ioProgrammo Anno XI - N°2 (111)

ANTEPRIMA: IBM DATASTAGE
IL SOFTWARE DI BIG BLUE PER TRASPORTARE I DATI IN QUALUNQUE FORMATO

VERSIONE PLUS RIVISTA+LIBRO+CD €9,90
VERSIONE STANDARD RIVISTA+CD €6,90

ioPROGRAMMO

PER ESPERTI E PRINCIPIANTI

Poste Italiane S.p.A Spedizione in A.P. • D.L. 353/2003 (conv.in L.27/02/2004 n.46) art.1 comma 2 DCB ROMA Periodicità mensile •-FEBBRAIO 2007-• ANNO XI, N.2 (111)

JAVA6 C'È
AGGIORNA SUBITO IL TUO CODICE

Migliora del 100% le tue applicazioni sfruttando tutte le caratteristiche della nuova versione

Desktop Oriented: scrivi software che si integra perfettamente con il sistema operativo

Integrato con Javascript: puoi richiamare i tuoi script direttamente dall'interno del codice

Controllo totale: grazie al debugger migliorato, al supporto per la sicurezza e alle "Compiler API"

Database ready: contiene "Derby" il nuovo DB embedded targato Apache

A PASSEGGIO CON SQL SERVER
Metti un database sul cellulare e sincronizzalo con un server centralizzato quando è possibile

C++ DIETRO LE QUINTE
Se il tuo programma va in "crash" potrebbe esserci una ragione nascosta. Noi ti mostriamo come realizzare software a prova d'errore...

MOBILE
METTI IL CATALOGO NEL CELLULARE
Con il nuovo Sun Java Toolkit lo fai in modo visuale con pochi semplici passi

VISUAL BASIC
GESTIRE FACILMENTE I FILE COMPRESSI
Ecco come creare, modificare, espandere file ZIP, RAR o CAB senza problemi

AJAX
UN NEWSTICKER PER IL TUO SITO
Facciamolo con Javascript, HTML e poco altro. Pesanti addon addio!

DATABASE
MYSQL UN SEGUGIO PER LA RICERCA
Trova quello che ti serve in un database gigante senza impiegarci un'eternità

REPORTISTICA SEMPLICE
Trasformiamo Active Reports in un potente server centralizzato che offre servizi a molti client

PATTERN
SOFTWARE BEN STRUTTURATO
I metodi template: "regole da seguire" per non rischiare di produrre codice spaghetti...

C++
FAI CONVIVERE C++ CON .NET
Codice completamente sotto controllo, ma senza rinunciare alle comodità di .NET

JAVA
ECCO L'EDITOR VISUALE PER JAVA
Disegnare le pagine Web non è mai stato così facile con Netbeans e le JSF

UN CLIENT FTP CON WEBWORKS
Arriva il framework che sostituisce Struts. Impariamo a usarlo con un esempio utile

SOLUZIONI Algoritmi evolutivi. Risolviamo problemi generici simulando il comportamento dell'uomo

70111
9 771128 594009

JAVA6 C'È • A PASSEGGIO CON SQL SERVER • C++ DIETRO LE QUINTE • METTI IL CATALOGO NEL CELLULARE • GESTIRE FACILMENTE I FILE COMPRESSI — EDIZIONI MASTER

Anno XI - N.ro 02 (111) - Febbraio 2007 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
Vice Publisher: Paolo Soldan
Redazione: Fabio Farnesi
Collaboratori: R. Allegra, L. Buono, A. Galeazzi, F. Grimaldi, F. Smelzo, A. Pelleriti, M. Locuratolo, L. Corias
Segreteria di Redazione: Veronica Longo

Realizzazione grafica: Cromatika S.r.l.
Art Director: Paolo Cristiano
Responsabile grafico di progetto Salvatore Vuono
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Francesco Cospite

Realizzazione Multimediale: SET S.r.l.
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via C. Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Presidente e Amministratore Delegato: Massimo Sesti
Direttore Generale: Massimo Rizzo

ABBONAMENTO E ARRETRATI

ITALIA: Abbonamento Annuale: ioProgrammo (11 numeri) €59,90 sconto 21% sul prezzo di copertina di €75,90 · ioProgrammo con Libro (11 numeri) €75,90 sconto 30% sul prezzo di copertina di €108,90 Offerte valide fino al 31/03/07 costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via C. Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito Visa, Cartasì, o Eurocard/Mastercard (inviando la Vs. autorizzazione, il numero di carta di credito, la data di scadenza, l'intestatario della carta e il codice CVV2, cioè le ultime 3 cifre del codice numerico riportato sul retro della carta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o BCC MEDIOCRATI S.C.AR.L. c/c 0 000 000 12000 ABI 07062 CAB 80880 CIN P (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: qualora nei prodotti fossero rinvenuti difetti o imperfezioni che ne limitassero la fruizione da parte dell'utente, è prevista la sostituzione gratuita, previo invio del materiale difettato.
La sostituzione sarà effettuata se il problema sarà riscontrato e segnalato entro e non oltre 10 giorni dalla data effettiva di acquisto in edicola e nei punti vendita autorizzati, facendo fede il timbro postale di restituzione del materiale.
Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master - Servizio Clienti - Via C. Correnti, 1 - 20123 Milano

Servizio Abbonati:
☎ tel. **02 831212**
@ e-mail: **servizioabbonati@edmaster.it**

Assistenza tecnica: ioprogrammo@edmaster.it
Stampa: Arti Grafiche Boccia S.p.a. Via Tiberio Felice, 7 Salerno
Stampa CD-Rom: Neotek S.r.l. - C.da Imperatore - Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Gennaio 2007

Questo mese su ioProgrammo

UN PANORAMA RICCO

Mai come in questo periodo la programmazione ha vissuto un momento di fervore così intenso. Da un lato l'arrivo di Ajax ha messo sottosopra il modo di sviluppare le applicazioni web. Dall'altro lato si affacciano sul mercato una serie di linguaggi incredibilmente validi. Contemporaneamente gli SDK più conosciuti sono stati aggiornati a versioni completamente rinnovate. Un primo elemento di innovazione l'aveva portato il .NET framework 3.0. Non abbiamo ancora digerito tutte le novità che quest'ultimo porta ed ecco che fa capolino sul mercato la nuova release del compilatore Java. In tutti e due i casi non si tratta di release di minore importanza ma di veri e propri elementi di discontinuità che introducono tecniche singolari e significative. Al contempo linguaggi come Python e Ruby che sembravano destinati a settori di nicchia arrivano invece sul mercato con un ricco carico di innovazione. Infine si registra movimento anche nel mercato degli IDE e degli strumenti di sviluppo. In questo settore la parte del leone l'ha fatta Borland con i suo nuovi strumenti dedicati alla produttività individuale. Tuttavia c'è da segnalare un'evoluzione molto interessante di Netbeans e la convergenza di Jbuilder 2007 verso Eclipse. Tutto questo fa presupporre che il mercato dello sviluppo in questo 2007 sarà particolarmente movimentato e che a noi programmatori non resta che aggiornarci continuamente per proporre ai nostri clienti soluzioni sempre tecnicamente valide, ma anche per abbattere i tempi interni di sviluppo.
Se fino a qualche tempo fa pensavamo che il mercato più importante su cui indirizzare la nostra attenzione fosse il mobile, adesso dobbiamo dire che non abbiamo che l'imbarazzo della scelta. La nostra unica preoccupazione deve essere quella di riuscire a stare al passo con i tempi

Fabio Farnesi
ffarnesi@edmaster.it

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre *\soft\codice* e *\soft\tools*) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.

JAVA6 C'È
Aggiorna subito il tuo codice

Migliora al 100% le tue applicazioni sfruttando tutte le caratteristiche della nuova versione

- ✔ **Desktop Oriented: scrivi software che si integra perfettamente con il sistema operativo**
- ✔ **Integrato con Javascript: puoi richiamare i tuoi script direttamente dall'interno del codice**
- ✔ **Controllo totale: grazie al debugger migliorato, al supporto per la sicurezza e alle "Compiler API"**
- ✔ **Database ready: contiene "Derby" il nuovo DB embedded targato Apache**

pag. 16

A PASSEGGIO CON SQL SERVER

Metti un database sul cellulare e sincronizzalo con un db centralizzato solo quando è possibile

QUALCHE CONSIGLIO UTILE

I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all'articolo e seguire passo passo quanto viene spiegato tenendo d'occhio l'intero progetto. Spesso per questioni di spazio non possiamo inserire il codice nella sua interezza nel corpo dell'articolo. Ci limitiamo a inserire le parti necessarie alla stretta comprensione della tecnica.

http://forum.ioprogrammo.it

Le versioni di ioProgrammo

RIVISTA + CD-ROM in edicola

JAVA 6 DEVELOPMENT KIT
La nuova versione del compilatore

Finalmente è arrivato. Ce ne parla Federico Paparoni nell'articolo di copertina di questo mese. Con le nuove estensioni per il Desktop e la maggiore integrazione con il sistema operativo d'origine la nuova versione di Java oltre che un'innovazione tecnologica fa segnare un pesante cambiamento di rotta di tipo filosofico, staremo a vedere quanto questa nuova direzione inciderà nel mercato dello sviluppo multipiattaforma. In ogni caso Java 6 rappresenta un grosso punto di disconuità rispetto al passato- I programmatori di Sun hanno fatto uno sforzo enorme per recuperare il gap in termini di prestazioni ed inserire nel linguaggio elementi che consentano di monitorare il flusso dell'applicazione in tempo reale e che non lascino il programmatore senza nessuna indicazione rispetto alla gestione della memoria e agli altri elementi di esecuzione del codice. Inoltre stanno incominciando a nascere strumenti di tipo visuale che consentono una maggiore rapidità di programmazione. sicuramente una migrazione dalle vecchie versioni è d'obbligo

Prodotti del mese

PHPBB 3.0B4

IL FORUM PER ECCELLENZA

Scritto in PHP con l'ausilio di MySQL è stata probabilmente la prima applicazione OpenSource a proporre un modello per lo sviluppo di forum sul Web. Attualmente è probabilmente il prodotto che espone il maggior numero di funzionalità, più facilmente estendibile e più ricco di moduli. In alcune versioni sono stati riscontrati problemi di sicurezza, tuttavia prontamente risolti. Le funzionalità sono infinite, tanto he nel tempo PHPBB è diventato un vero e proprio standard e sono molte le applicazioni concorrenti che ne inseguono le capacità. Se avete bisogno di installare un forum o più semplicemente volete studiare alcune tecniche per scrivere codice elegante e funzionale in PHP, probabilmente te PHPbb è il software che fa per voi

[pag.106]

Zend Google Data 0.6.0

Il framework per scrivere sul web

Google recentemente sta ampliando la sua gamma di prodotti proponendo agli sviluppatori una serie di API che consentono di interfacciare i propri programmi con i servizi messi a disposizione dal motore di ricerca. Fra le API più recenti compare questa Google Data, ovvero un protocollo che consente di scambiare i dati secondo regole ben precise sul Web. Zend ha immediatamente colto la palla al balzo proponendo questo interessante framework che rende più immediato l'interfacciamento di questo protocollo con le applicazioni, ovviamente, PHP. Non mancheremo di approfondire questo argomento nei prossimi numeri di ioProgrammo

[pag.109]

IronPython 1.0.1

Un'implemntazione di Python in tecnologia .NET

In molti conoscono Python. Si tratta di un linguaggio agile, dinamico, elegante, ad oggetti e multipiattaforma. Per le sue caratteristiche di estrema flessibilità, per la sua facilità nella curva di apprendimento, per la sua leggerezza si tratta di un linguaggio estremamente diffuso su piattaforma Linux dove viene utilizzato per automatizzare gran parte delle impostazioni di sistema. Anche sul Web Python ha trovato una sua collocazione ben precisa, pare infatti che sia il linguaggio di scripting su cui si basa buona parte di Google. In ambiente Windows si sta diffondendo a macchia d'olio, tanto che ne è appena nata anche una versione per .NET, appunto IronPython. Le caratteristiche sono identiche a quelle classiche di Python, ma il codice prima di essere eseguito viene convertito nel MIL di .NET

[pag.109]

Tomcat 6.0.2

L'application server per JSP

Se siete dei programmatori JSP avete senza dubbio bisogno di Tomcat per testare e utilizzare le vostre applicazioni. Ne esistono tanti, da JBOSS a JRUN, ma Tomcat per la sua leggerezza e completezza è uno dei più diffusi. Prima di tutto è senza dubbio un server web ovvero può funzionare da server web ma è soprattutto un Application Server che vi consente di utilizzare la tecnica delle servlet o delle JSP per creare applicazioni Web con una logica business e con il linguaggio Java a fare da asse portante. Tomcat ha in se anche un Web Server, ma quel che più conta è un container di applicazioni Java. Potete e dovete utilizzarlo se volete imparare a programmare per il Web utilizzando il linguaggio di Sun. In questo numero viene utilizzato per testare l'articolo relativo allo sviluppo con JSF

[pag.109]

I contenuti del libro

Programmare con AJAX

Ajax è celebrato da più parti come una delle novità più eclatanti degli ultimi anni. È grazie ad Ajax che oggi si parla con sempre maggiore convinzione del Web 2.0. Ma di cosa si tratta? sostanzialmente di una tecnologia che consente di gestire le applicazioni web con una modalità simile a quella delle applicazioni desktop, ovvero evitando il reload delle pagine. Sembrerebbe una novità da poco, tuttavia consente di creare un punto di convergenza tra le enormi possibilità offerte dalle applicazioni Internet e quelle delle applicazioni desktop, rendendo il web decisamente più comodo da utilizzare. A tutto ciò si deve aggiungere che Ajax non è una tecnologia complessa da apprendere, per tale motivo rappresenta una novità di assoluta importanza. Francesco Smelzo ci descrive i vari aspetti di Ajax con un linguaggio semplice e corredando gli aspetti teorici con una miriade di esempi che ci conducono passo passo nell'apprendimento di una delle tecniche più importanti per il futuro del Web

LA GUIDA PER PORTARE FACILMENTE LE TUE APPLICAZIONI NEL MONDO DEL WEB 2.0

- **Teoria e tecnica alle basi di Ajax**
- **Uno sguardo a DOM HTML**
- **XML e la gestione dei dati**
- **Ajax on the Road**

GLI ALLEGATI DI IOPROGRAMMO

Questo mese tre webcast dedicati agli "architetti" del software, coloro che progettano e disegnano soluzioni

Architettura del software: un'introduzione

Nel corso di questo webcast che fa parte della serie "Architettura del software: nozioni introduttive" viene presentato il concetto di "architettura" del software e delle soluzioni, definendo il ruolo di un architetto all'interno del team di progetto.
Speaker: Andrea Saltarello.

Architettura del software: dai requisiti ai casi d'uso

La raccolta dei requisiti è il primo passo verso la definizione dell'architettura di una soluzione: nel corso di questo webcast introdurremo il concetto di "caso d'uso" e ne spiegheremo il ruolo all'interno del processo di design del software. Saranno inoltre forniti i concetti di UML (Unified Modeling Language) propedeutici alla formalizzazione dei casi d'uso.
Speaker: Lorenzo Barbieri

Architettura del software: dai casi d'uso al modello

Formalizzati i casi d'uso, è possibile derivarne il diagramma di struttura statica, ossia il modello contenente la definizione strutturale del sistema. Durante il webcast saranno forniti i concetti di UML (Unified Modeling Language) propedeutici alla modellazione di un diagramma di struttura statica. Le nozioni attinenti i casi d'uso sono invece considerate un prerequisito acquisito mediante il webcast Architettura del software: dai requisiti ai casi d'uso.
Speaker: Lorenzo Barbieri.

msdn WEBCAST

PERCORSI FORMATIVI

Per chi desidera approfondire sono disponibili sul sito di Microsoft una serie di strumenti dedicati agli Architetti

• Domain Driven Design: Overview (Livello 300) • UML Reloaded (Livello 300) • Architettura del software: .NET e gli strumenti • Architettura del software: disegno architetturale, gli idiomi e le linee guida di design per .NET Framework • Architettura del software: pattern by example • Architettura del software: introduzione ai design pattern • Architettura del software: un approccio agile

Visita
http://www.microsoft.com/italy/msdn/risorsemsdn/architetti/aspire.mspx

FAQ

Cosa sono i Webcast MSDN?

MSDN propone agli sviluppatori una serie di eventi gratuiti online e interattivi che approfondiscono le principali tematiche relative allo sviluppo di applicazioni su tecnologia Microsoft. Questa serie di "corsi" sono noti con il nome di Webcast MSDN

Come è composto tipicamente un Webcast?

Normalmente vengono illustrate una serie di Slide commentate da un relatore. A supporto di queste presentazioni vengono inserite delle Demo in presa diretta che mostrano dal vivo come usare gli strumenti oggetto del Webcast

Come mai trovo riferimenti a chat o a strumenti che non ho disponibili nei Webcast allegati alla rivista?

La natura dei Webcast è quella di essere seguiti OnLine in tempo reale. Durante queste presentazioni in diretta vengono utilizzati strumenti molto simili a quelli della formazione a distanza. In questa ottica è possibile porre domande in presa diretta al relatore oppure partecipare a sondaggi etc. I Webcast riprodotti nel CD di ioProgrammo, pur non perdendo nessun contenuto informativo, per la natura asincrona del supporto non possono godere dell'interazione diretta con il relatore.

Come mai trovo i Webcast su ioProgrammo

Come sempre ioProgrammo cerca di fornire un servizio ai programmatori italiani. Abbiamo pensato che poter usufruire dei Webcast MSDN direttamente da CD rappresentasse un ottimo modo di formarsi comodamente a casa e nei tempi desiderati. Lo scopo tanto di ioProgrammo, quanto di Microsoft è infatti quello di supportare la comunità dei programmatori italiani con tutti gli strumenti possibili.

Su ioProgrammo troverò tutti Webcast di Microsoft?

Ne troverai sicuramente una buona parte, tuttavia per loro natura i webcast di Microsoft vengono diffusi anche OnLine e possono essere seguiti previa iscrizione. L'indirizzo per saperne di più è: http://www.microsoft.it/msdn/webcast/msdn segnalo nei tuoi bookmark. Non puoi mancare.

L'iniziativa sarà ripetuta sui prossimi numeri?

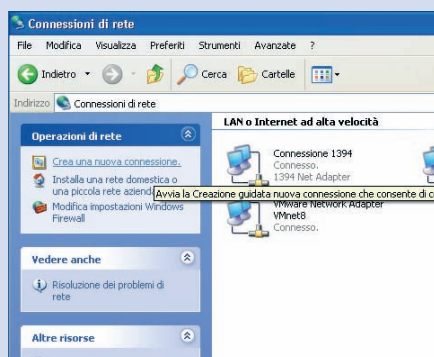
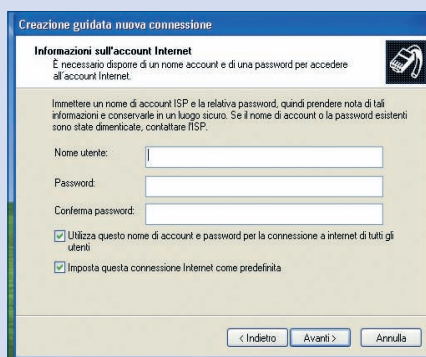
Sicuramente si.

Online con Tiscali Easy

Numero unico per tutta l'Italia e nessuna registrazione. Il modo più semplice e veloce per entrare in Internet risparmiando

Sei spesso lontano da casa e hai necessità di collegarti a Internet ovunque ti trovi? Desideri salvaguardare la tua privacy navigando in modo anonimo? Non hai voglia di perdere tempo in lunghe registrazioni per creare un nuovo account? Niente paura, Tiscali ha pensato anche a te! Con Tiscali Easy arriva un sistema tutto nuovo di collegarsi a Internet. Non sarà più necessario creare un nuovo account e fornire i propri dati personali, potrai navigare collegandoti con un unico numero di telefono (7023456789) da tutta Italia e soprattutto utilizzando i dati di accesso forniti direttamente da Tiscali (UserID e Password presenti sulla scheda), quindi non collegabili in alcun modo a te. Insomma, con Tiscali Easy, otterrai in un colpo solo riservatezza dei dati, risparmio del tempo necessario alla creazione di un abbonamento e tariffe vantaggiose. I costi di connessione sono simili a quelli di un classico abbonamento gratuito: 1,90 cent. per i primi 10 minuti e 1,72 cent. per quelli successivi. Per risparmiare ulteriormente è possibile connettersi a Internet durante le ore serali o nei giorni festivi al costo di 1,09 cent. per i primi dieci minuti e 0,98 cent. per quelli successivi. L'unico requisito richiesto per poter eseguire la connessione è un modem correttamente installato. Poiché si tratta di una normale connessione analogica è possibile utilizzare i tool di accesso remoto integrati in Windows

IN RETE CON TISCALI

Nessuna registrazione basta creare una nuova connessione e inserire i dati di accesso forniti da Tiscali

1 NUOVA CONNESSIONE
Portiamoci nel pannello di controllo, e selezioniamo l'icona connessioni di rete. In alto a sinistra selezioniamo "nuova connessione" e prepariamoci a seguire il wizard che comparirà

2 DATI DI ACCESSO
Seguiamo il Wizard fino a quando non ci vengono chiesti i dati per l'autenticazione. E' sufficiente inserire quelli presenti nella card allegata a questo numero di ioProgrammo: master2007/tiscali

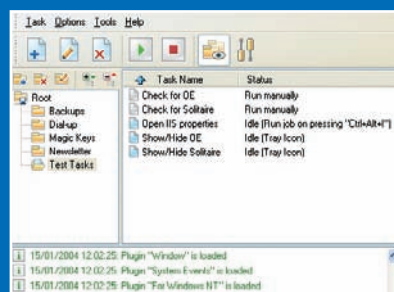
3 ACCESSO A INTERNET
Connettersi è semplicissimo, troveremo una nuova icona all'interno del pannello di controllo alla voce connessioni. Bastano due click ed il gioco è fatto sarete connessi ovunque vi troviate

News

FIREFOX 3.0 QUASI PRONTO

Sono in molti a credere che questa release del browser antagonista di Internet Explorer segnerà l'inizio di una rivoluzione per quanto riguarda questo tipo di applicativi. Già in passato in molti avevano dovuto rincorrere le novità tecnologiche di Firefox, e anche questa volta ci sono motivi per credere che questa nuova release farà da apripista ad una serie di innovazioni importanti. Si va dal supporto alle librerie Cairo e dunque alla piena compatibilità con SVG, all'auspicata uniformità di visualizzazione su tutte le piattaforme. Firefox 3 è quasi pronto ed è già possibile scaricarne una beta all'indirizzo *http://wiki.mozilla.org/Firefox3*

AUTOMAZIONE AL TOP CON ROBOTASK

È appena stata rilasciata la versione 2.5 e si incominciano a registare i primi segni di apprezzamento per un software che consente di svincolarsi dalle operazioni ripetitive e permette di creare task da far svolgere al sistema in momenti particolari evitando di dover scrivere noiosi script. Un bell'esempio di programma intelligente e semplice che senza troppo "chiasso" può aiutare il nostro lavoro

PRONTE LE PATCH PER JAVA

Sun ha appena rilasciato la sua versione numero 6 del compilatore per Java e contemporaneamente si è preoccupata di riparare alcune falle anche gravi presenti nelle precedenti versioni.
In particolare sarebbero 7 le gravi vulnerabilità presenti nelle versioni 1.5.x, 1.4.x e 1.3.x della macchina virtuale. La notizia buona è che con le nuove patch le falle sono state corrette, quella cattiva è che fino ad ora siamo stati a rischio di attacchi piuttosto gravi. Secondo quanto dichiarato da FrSIRT le vulnerabilità in questione sarebbero state tali da consentire addirittura ad un eventuale aggressore di prendere il pieno possesso del sistema.
Fra le sette debolezze maligne ne compaiono due che coinvolgerebbero la lettura/scrittura di informaziomnni sul file system. Le altre falle riguarderebbero la gestione di immagini di grandi dimensioni, di array e di valori negativi. Sfruttando questi tre "punti di ingresso" sarebbe possibile far eseguire del codice alla JVM. Infine un paio di vulnerabilità riguardano la possibilità di diffonfere informazioni personali presenti nel sistema attaccato.
Le nuove versioni sono disponibili sul sito di Sun. Anche se la miglior soluzione possibile è quella di passare alla nuova versione 6 che almeno nelle dichiarazioni di Sun, nei primi commenti, e nei nostri test per il momento è dichiarata abbastanza sicura

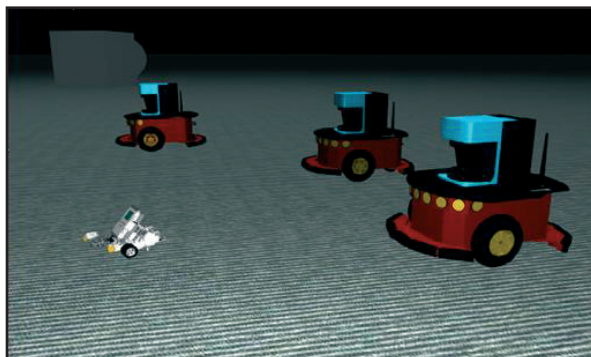
DISPONIBILE L'SP1 PER VISUAL STUDIO

Sono oltre 70 le migliorie apportate a Visual Studio con questo suo SP1, secondo quanto dichiarato da Microsoft. La totalità o quasi delle modifiche sarebbe stata possibile grazie alle varie segnalazioni effettuate dagli utenti. Si tratta di modifiche di varia entità, alcune delle quali migliorerebbero funzionalità già esistenti come la colorazione di parti di codice altre sarebbero più significative fino a giungere a quelle che coinvolgono il miglioramento delle performance e della stabilità del prodotto.
Purtroppo non è ancora disponibile il SP1 per Visual Studio in ambiente Vista. Di fatto chi proverà ad installare lo strumento principe per lo sviluppo sul nuovo sistema operativo si troverà di fronte ad un fastidioso "compatibility issue". In ogni caso Microsoft sta lavorando anche al rilascio di questa patch che dovrebbe rendere Visual Studio 2005 pienamente compatibile con Vista e di conseguenza pronto per il .NET 3.0 senza bisogno di ulteriori installazioni

ITALIA DISCONESSA

L'ISTAT ha appena reso noti i risultati sull'indagine relativa alla diffuzione della tecnologia nelle famiglie. Il panorama che ne è emerso non è roseo per l'Italia. È vero che siamo passati dall'11.6 per cento al 14.4 per cento per quanto riguarda la diffusione della banda larga, tuttavia è ancora troppo poco per poter competere con gli altri paesi europei. Se parliamo di connettività in generale l'italia ottiene un 40% a confronto con una media europea del 52%, se poi volessimo rapportarci ad esempio all'Olanda, ci scontremmo con un significativo 80%. Il nostro paese sopravanza di poco la Spagna con il 39% e il Portogallo con un 35%, poco sopra di noi si pone la Francia con un 41%.
Gli Italiani si confermano come un popolo tradizionalista. Internet e la connetività in genere sono sentite come attività necessarie allo svolgimento del proprio lavoro ma non ancora come migliorie alla qualità della vita anche in famiglia. Di fatto nelle imprese il tasso di connetività sale al 72.2%, dato che non trova riscontro nelle connessioni di tipo familiare. Alla domanda relativa ai motivi di "mancata connetività" per le famiglie, il 39,6% ha dichiarato di non essere interessata. Mettere in relazione i due dati è facile: gli italiani usano internet in ufficio, ma a casa preferiscono ancora il comodo divano e la buona vecchia televisione. Buoni i risultati ottenuti dalla PA, sale infatti al 26,6% l'utilizzo dei servizi di modulistica OnLine messi a disposizione dalla pubblica amministrazione. Infine è da segnalare un 20.6% di famiglie italiane che utilizzano in una qualche misura i servizi di ecommerce. Purtroppo permane una forbice notevole fra Nord e Sud per quanto riguarda ciascuna di queste statistiche

ARRIVA IL MICROSOFT ROBOTICS SDK

Per coloro che pensavano che il limite fra scienza e fantascienza fosse ancora distante arriva una reale e tangibile smentita dal mondo dell'informatica, Microsoft ha appena presentato un ambiente di sviluppo in grado di programmare il proprio robot. Si tratta di una piattaforma piuttosto sofisticata che si può interfacciare con vari tipi di soluzioni elettroniche. All'interno in ogni caso è presente un ambiente virtuale tramite il quale è possibile testare la validità dei propri progetti. Secondo Microsoft il solo ambiente dovrebbe essere sufficiente anche a chi non ha una vasta esperienza di programmazione alla creazione e modellazione del proprio robot. L'ipotesi di Microsoft è consentire lo sviluppo tramite semplice Drag & Drop. I linguaggi supportati sono tutti quelli della piattaforma .NET ma desta una certa sorpresa anche la presenza di IronPython, ovvero il porting del famoso linguaggio OpenSource su piattaforma Microsoft.
Il costo del prodotto è di appena 399$ per chi desiderasse farne un uso commerciale viceversa è gratuito per usi domestici

GOOGLE OPENSOURCE

Il gigante dei motori di ricerca ha reso disponibile l'intero codice sorgente del proprio Google Web Toolkit. Si tratta di uno strumento che Google aveva messo a disposizione degli sviluppatori per la creazione rapida di web application in modalità Ajax.
Il passo è di quelli importanti e mostra una certa sensibilità da parte di Google verso il mondo dello sviluppo. D'altra parte non sono poche le API e le interfacce che google ha messo a disposizione degli sviluppatori per accedere ai propri servizi. La disponibilità del codice sorgente del Google Web Toolkit rappresenta un'ottima occasione per comprendere a fondo come Google interpreta la tecnologia Ajax oltre che per modificare a proprio piacimento le funzionalità del toolkit stesso. Annotiamo che Google è stata una delle prime aziende ad adottare Ajax

JAVA 6 E' ARRIVATO MIGRA IL TUO CODICE

IL NUOVO COMPILATORE È RICCO DI NOVITÀ. ECCO LE COSE DA SAPERE PER AGGIORNARE SUBITO IL TUO SOFTWARE E SFRUTTARE TUTTE LE CARATTERISTICHE DELLA NUOVA VERSIONE. PIÙ PERFORMANCE PIÙ INTEGRAZIONE CON IL DESKTOP

Da molto tempo si parlava del rilascio della versione 6 del compilatore Java prodotto da Sun. La versione finale è ora disponibile, non rimane che da capire cosa cambia rispetto al compilatore precedente e come e perché prepararsi alla migrazione. In questo articolo introdurremo le feature che sono state introdotte in Java 6, nome in codice "Mustang" e soprattutto vedremo come riadattare il vecchio codice delle nostre applicazioni per renderlo compatibile con la nuova piattaforma. dotte in questa versione, cercano di colmare un vuoto presente da ormai troppo tempo per quanto riguarda Java, rispetto ad altri linguaggi concorrenti (.NET in primis). Andiamo ora a considerare tutti quei cambiamenti necessari al nostro codice per renderlo compatibile con la nuova versione. In seguito vedremo come poter cambiare le nostre applicazioni per utilizzare le nuove feature che sono state messe a disposizione.

NOVITÀ PUNTO PER PUNTO

Prima di ogni cosa riportiamo brevemente sottoforma di sommario un elenco breve delle novità che sono state introdotte. Nel corso dell'articolo ne alizzeremo molte nel dettaglio:

- Nuove API per lo sviluppo di Web Services (digital signature, JAX-WS e JAXB)
- Approvazione della JSR 223 ed introduzione del supporto per lo scripting (Javascript ed altri).
- Miglioramenti sulle API di sicurezza ed introduzione di nuove API riguardanti Smart Card, Kerberos, LDAP
- Introduzione di JDBC 4.0
- Miglioramenti sulle API riguardanti JMX
- Miglioramenti su memory management
- Introduzione di nuove ed interessanti librerie standard
- Miglioramenti sulla parte Desktop, sull'integrazione con sistema operativo e sulle interfacce utente

Una delle novità che ha fatto più scalpore è stata quella riguardante la parte Desktop. Infatti a partire da questa versione abbiamo dei significativi cambiamenti che sembrano indicare un riavvicinamento del linguaggio Java al mondo Desktop. Le opinioni degli sviluppatori su internet infatti sono concordi nel dire che le feature relative al Desktop intro-

COMPATIBILITÀ

Come in ogni passaggio di versione ci sono una serie di cambiamenti nelle API che devono essere presi in considerazione. Il passaggio dalla versione 1.4.2 alla versione 5 aggiunse al linguaggio diversi costrutti che prima non erano presenti (generics, foreach etc. etc.). Le "deprecated" genereranno a tempo di compilazione il classico warning *"Note: prova.java uses a deprecated API. Please consult the documentation for a better alternative"*. Questa versione chiaramente è compatibile dal punto di vista binario con la precedente (5.0), fatta eccezione per una serie di incompatibilità delle quali riportiamo un breve riassunto:

- La retrocompatibilità binaria dei programmi java è garantita per quanto riguarda i programmi generati con il compilatore standard javac. Se vengono utilizzati altri tool per la compilazione o per l'offuscamento del codice è possibile che non siano ancora stati aggiornati e che quindi non conoscano il nuovo formato utilizzato.
- La classe FileLock del package java.nio.channels controlla i file che sono già stati "lockati" da altre istanze di FileChannel. Nella precedente versione non c'era questo tipo di controllo e poteva succedere che un istanza di FileChannel potesse interagire con un file utilizzato da un'altra istanza. In questo versione è stato introdotto questo controllo e la relativa eccezione, OverlappingFile LockException, nel caso in cui non sia possibile

utilizzare il file. Sempre in relazione a questa nuova feature è stata aggiunta una nuova proprietà, sun.nio.ch.disableSystemWideOverlapping FileLockCheck che permette di abilitare o meno questo tipo di controllo.

- È stato rafforzato il controllo sui cast nel momento di compilazione. Praticamente potrebbe succedere che dei cast, che comunque non andrebbero bene, possono essere individuati al momento della compilazione da javac.
- I programmi che richiedevano l'esplicito utilizzo da linea di comando di -Xbootclasspath:<path to rt.jar> adesso non funzioneranno perché queste risorse si trovano ora in un jar differente di nome resources.jar.
- Per migliorare le prestazioni grafiche della applicazioni Swing è stato introdotto un implementazione completa del double buffering. Praticamente per ogni componente grafico che un buffer che viene utilizzato per velocizzare la visualizzazione dei componenti a schermo. Questo viene reso possibile dalla classe RepaintManager del package javax.swing. Le classi grafiche che all'interno di un programma non utilizzano questo meccanismo del double buffering potrebbe andare incontro a dei problemi di lentezza.
- È stata cambiata l'implementazione del Drag & Drop rispetto alle precedenti versioni. Prima quando dovevamo trascinare un oggetto all'interno di componenti Swing dovevamo prima selezionarlo e successivamente trascinarlo, quindi operare due diversi click con il mouse. In questa nuova versione è stato implementato il Drag & Drop senza dover cliccare due volte, quindi questo ha una serie di ripercussioni sugli eventi relativi ai MouseListener, JTable e JTree. C'è bisogno quindi di un cambiamento nella logica di gestione degli eventi relativi a queste interfacce grafiche.

Questa è solo una parte delle novità che sono state introdotte nella nuova versione cambiando vecchie API. Per un maggiore approfondimento vi rimando alla pagina di compatibilità della versione 6, http://java.sun.com/javase/6/webnotes/compatibility.html , e alla documentazione javadoc relativa alle nuove API.

MONITORAGGIO E PERFORMANCE

I programmi che scriviamo devono chiaramente sottostare a certi limiti per quanto riguarda le performance. Non possiamo pensare di poter eseguire le nostre applicazioni su macchine con 8 processori e svariati giga di RAM, quindi una parte importante è quella riguardante le performance. Rispetto alle versioni precedenti di Java abbiamo ora a disposizione diversi nuovi strumenti da utilizzare per ottimizzare le nostre applicazioni e capire quali sono le parti del nostro codice che creano dei problemi. Già a partire dalla versione 5.0 sono stati introdotti dei strumenti per gli sviluppatori per monitorare i propri programmi e rendersi meglio conto di quali sono le performance. Questi tool nella versione 6 hanno raggiunto una certa maturità e qui riportiamo una descrizione per l'utilizzo

- JSTAT (JVM Statistics Monitoring Tool): Si connette ad una JVM e permette di loggare informazioni e statistiche. Qui di seguito viene riportato un esempio di JSTAT da linea di comando.

```
C:\Programmi\Java\jdk1.6.0\bin>jps
2036
1868 Main
1996 Jps

C:\Programmi\Java\jdk1.6.0\bin>jstat -class 1868
Loaded  Bytes  Unloaded  Bytes     Time
  4805  5530,0        0    0,0    10,53
```

- JSTATD : Un server RMI che permette di avere informazioni sulle JVM in esecuzione e permette inoltre un monitoraggio remoto
- JINFO: Permette di avere informazioni riguardanti un singolo processo Java. Per avere la lista di processi java è possibile utilizzare un ulteriore tool da riga di comando, jsp, che elenca gli identificativi dei processi java.
- JMAP: Fornisce informazioni riguardanti gli oggetti di memoria e la memoria heap utilizzati da un processo. Permette anche di produrre come output un dump dell'heap, che può essere utilizzato successivamente per essere analizzato.
- JHAT: E' un webserver che permette da browser di visualizzare informazioni relative ad un file di dump dell'heap. Molto utile per capire cosa succede di sbagliato in un'applicazione
- JSTACK: Fornisce lo stack trace di un processo Java. Qui di seguito viene riportato un esempio di JSTACK su un applet lanciata dal browser

```
C:\Programmi\Java\jdk1.6.0\bin>jps
2036
2124 Jps

C:\Programmi\Java\jdk1.6.0\bin>jstack 2036
2006-12-24 12:49:12
Full thread dump Java HotSpot(TM) Client VM
                    (1.6.0-b105 mixed mode, sharing):
"Thread-9" prio=2 tid=0x04f16800 nid=0x4f4 in
                    Object.wait() [0x0738f000..0x0738f
d94]
   java.lang.Thread.State: WAITING (on object
```

NOTE

COME SAPERNE DI PIÙ SU JTOP

In rete è possibile consultare un bell'articolo di Mandy Chung dove viene analizzata l'applicazione JTop e le nuove API per il monitoraggio e la gestione. Il link è http://weblogs.java.net/blog/mandychung/archive/2006/05/mustang_jconsol_1.html

```
                                             monitor)
        at java.lang.Object.wait(Native Method)
        at java.lang.Object.wait(Object.java:485)
        at Animator.run(Animator.java:358)
        - locked <0x210fcb68> (a Animator)
        at java.lang.Thread.run(Unknown Source)

"Java Sound Event Dispatcher" daemon prio=4
                 tid=0x04f0d400 nid=0x6a0 in Object.w
ait() [0x06d7f000..0x06d7fb14]
   java.lang.Thread.State: WAITING (on object
                                             monitor)
        at java.lang.Object.wait(Native Method)
        at java.lang.Object.wait(Object.java:485)
                              at com.sun.media.sound.
                          EventDispatcher.dispatchEvents
                                        (Unknown Source)
        - locked <0x211fe850> (a
                  com.sun.media.sound.EventDispatcher)
                                     at com.sun.media
                              .sound.EventDispatcher.run
                                        (Unknown Source)
        at java.lang.Thread.run(Unknown Source)
```

Oltre a tutti questi tool da riga di commando che abbiamo elencato, esiste un interessante strumento, JConsole, che ha fatto la sua comparsa già dalla versione 5.0 e che nella nuova versione di Java ha acquisito maggiore potenza. Per avviare JConsole basta digitare da linea di comando "jconsole" e una volta partita l'interfaccia grafica possiamo iniziare a monitorare JConsole stesso. All'inizio ci viene fatto scegliere quale processo vogliamo monitorare.

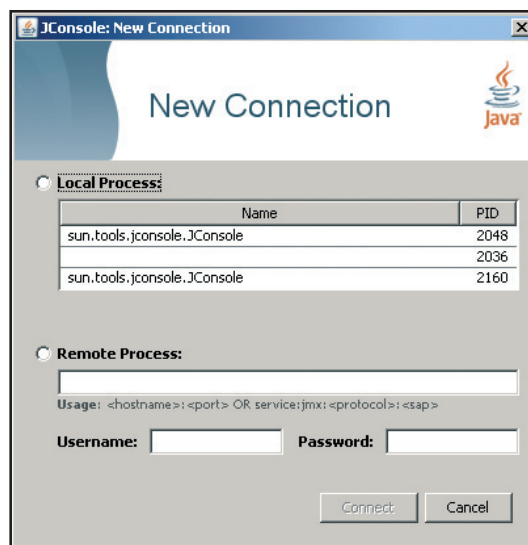



Fig. 1: ***Avvio di JConsole***

Dopo aver scelto il processo viene inizializzata l'intefaccia, mostrando quattro grafici che riassumono la nostra applicazione: utilizzo di memoria heap, numero di thread, classi e percentuale di CPU utilizzata.

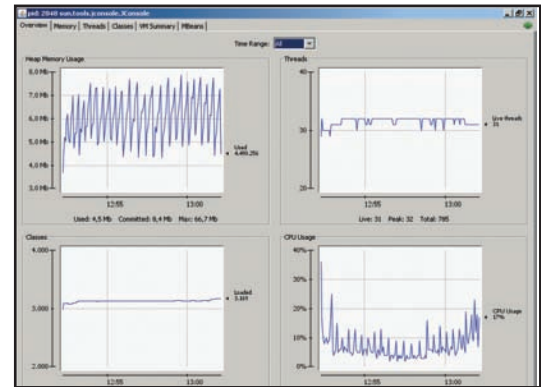



Fig. 2: ***Interfaccia di default per JConsole***

In Java 6 sono state ampliate le possibilità per gli sviluppatori, per monitorare meglio le proprie applicazioni. E' infatti possibile attraverso JMX creare dei plugin per JConsole, oppure delle vere e proprie applicazioni separate che possono monitorare aspetti particolari della memoria che ci interessano. Il seguente codice ad esempio ci permette di visualizzare informazioni sui Thread attivi

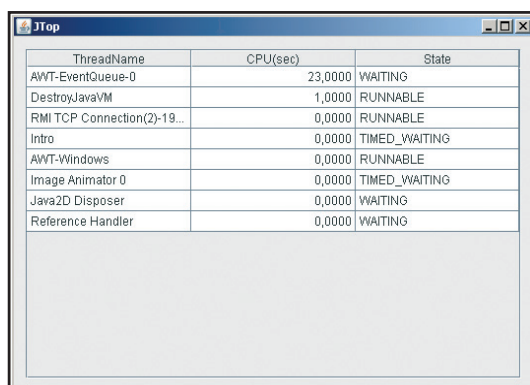
```
MBeanServerConnection server;
ThreadMXBean tmbean;
String urlPath = "/jndi/rmi://" + hostname + ":" +
                                      port + "/jmxrmi";
MBeanServerConnection server = null;
try {
           JMXServiceURL url = new
                   JMXServiceURL("rmi", "", 0, urlPath);
           JMXConnector jmxc =
                    JMXConnectorFactory.connect(url);
           server = jmxc.getMBeanServerConnection();
           tmbean = newPlatformMXBeanProxy(server,
THREAD_MXBEAN_NAME,
ThreadMXBean.class);
}catch (Exception e) {
           System.err.println("Exception: " +
                                     e.getMessage());
}
long[] tids = tmbean.getAllThreadIds();
ThreadInfo[] tinfos = tmbean.getThreadInfo(tids);
// build a map with key = CPU time and value =
                                          ThreadInfo
SortedMap<Long, ThreadInfo> map = new
                      TreeMap<Long, ThreadInfo>();
for (int i = 0; i < tids.length; i++) {
           long cpuTime =
                  tmbean.getThreadCpuTime(tids[i]);
           if (cpuTime != -1 && tinfos[i] != null) {
System.out.println("Thread name
                     :"+tids[i].getThreadName());
System.out.println("Thread id
                         :"+tids[i].getThreadId());
System.out.println("Thread suspended:
                           "tids[i].isSuspended());
                              System.out.println("CPU
```

RIFERIMENTI UFFICIALI SU JMX

La documentazione ufficiale SUN sulle nuove feature riguardanti il monitoraggio e la gestione delle applicazioni tramite JMX è reperibile all'indirizzo http://java.sun.com/javase/6/docs/technotes/guides/management/

```
Time: "+cpuTime);
        }
}
```

Questo codice non fa altro che collegarsi ad un server JMX che fornisce le informazioni e listarle a schermo. Le informazioni riguardano i thread, con il nome, l'id, la sospensione e il tempo di cpu. Per vedere un esempio funzionante possiamo utilizzare una delle demo presenti all'interno della release di Java 6. Dentro la cartella %JDK_HOME%/demo/management troviamo JTop, un codice d'esempio fornito da Mandy Chung, dove possiamo vedere il funzionamento di queste API. Per avviare l'esempio dobbiamo prima di tutto far partire un'applicazione che esponga l'agent JMX.

```
java -Dcom.sun.management.jmxremote.port=1090
        -
        Dcom.sun.management.jmxremote.ssl=false
        -
        Dcom.sun.management.jmxremote.authenticate
        =false
        -jar
<JDK_HOME>/demo/jfc/Java2D/Java2Demo.jar
```

In questo modo abbiamo fatto partire l'applicazione Java2Demo con l'agent JMX. Ora possiamo far partire JTop facendolo collegare a questo agent, anche da una postazione remota

```
java -jar
        <JDK_HOME>/demo/management/JTop/JTop.jar
        <hostname>:1090
```

Il risultato è un'applicazione grafica che ci permette di monitorare tranquillamente un'altra applicazione, come potete vedere dall'immagine riportata.

JTop

ThreadName	CPU(sec)	State
AWT-EventQueue-0	23,0000	WAITING
DestroyJavaVM	1,0000	RUNNABLE
RMI TCP Connection(2)-19...	0,0000	RUNNABLE
Intro	0,0000	TIMED_WAITING
AWT-Windows	0,0000	RUNNABLE
Image Animator 0	0,0000	TIMED_WAITING
Java2D Disposer	0,0000	WAITING
Reference Handler	0,0000	WAITING

Fig. 3: ***Applicazione JTop per il monotoraggio***

Per ulteriori approfondimenti su questo tema vi rimando alla documentazione ufficiale delle java.lang.management API, dove potrete trovare tutte le classi e metodi da utilizzare per realizzare qualche interessante programma di monitoraggio.

SCRIPTING PER JAVA

Una delle qualità che permette ad un linguaggio di scripting di diventare famoso è la facilità d'utilizzo e l'immediatezza nella risoluzione di algoritmi. Molti linguaggi di scripting devono la loro celebrità proprio a questo. Java dal lato suo, già con l'introduzione di EL, Expression Language, un linguaggio di scripting semplice da utilizzare all'interno di pagine web JSP, aveva dato la possibilità di "scriptare" all'interno di Java. Nella nuova versione di Java è stata inclusa la JSR 233, che permette agli sviluppatori di utilizzare linguaggi di scripting all'interno di programmi Java. La cosa può sembrare un po' strana, ma vediamo un attimo cosa significa tutto ciò. La JSR 233 definisce una serie di API che permettono allo sviluppatore di richiamare all'interno di un programma Java un interprete per un certo linguaggio di scripting e fargli eseguire un vero e proprio script. Queste permette, dal punto di vista del programmatore, di poter riutilizzare codice di altri linguaggi oppure di risolvere dei problemi utilizzando una via diversa da quella che propone Java. Tutto ciò aumenta le possibilità dal punto di vista dello sviluppatore per poter risolvere problemi. Vediamo qui di seguito un primo esempio di quello stiamo dicendo

```
import javax.script.*;
public class TestJavaScript {
        public static void main(String a[]){
                ScriptEngineManager
                mgr = new ScriptEngineManager();
                ScriptEngine jsEngine =
                mgr.getEngineByName("JavaScript");
                try {
                jsEngine.eval("print('Ciao da
                JavaScript dentro Java :P')");
                } catch (ScriptException
                ex) {
ex.printStackTrace();
                }
        }
}
```

Come prima cosa abbiamo istanziato uno ScriptEngineManager, la classe che ci permette di caricare all'interno del nostro programma un engine di scripting supportato dalla piattaforma che stiamo utilizzando. Successivamente carichiamo l'engine JavaScript utilizzando il metodo getEngineByName() di ScriptEngineManager. Infine invochiamo su questo engine uno script JavaScript, che stampa a schermo una stringa. Un'altra cosa interessante che riguarda la JSR233 è la definizione dell'interfaccia Invocable. Utilizzando questa interfaccia possiamo invocare i metodi definiti all'interno dello script, passando come variabili quelle presenti all'interno del nostro programma Java. Qui di

JAVASCRIPT E MOZILLA

E' possibile consultare lo standard di riferimento per l'implementazione di JavaScript in Java 6 all'indirizzo http://www.mozilla.org/rhino/

NOTE

RIFERIMENTI SULLO SCRIPTING IN JAVA6

La documentazione ufficiale è riportata all'indirizzo
http://java.sun.com/javase/6/docs/technotes/guides/scripting/index.html
http://java.sun.com/developer/technicalArticles/J2SE/Desktop/scripting/DesktopScripting.zip

seguito viene illustrato come dobbiamo fare per poter passare allo script i nostri valori

```
import javax.script.*;

public class TestInvocable {
    public static void main(String[] args) throws
                                    Exception {
        ScriptEngineManager manager = new
                          ScriptEngineManager();
        ScriptEngine engine =
              manager.getEngineByName("JavaScript");
        String script = "function saluta(name) {
                       print('Ciao, ' + name); }";
        engine.eval(script);

        Invocable inv = (Invocable) engine;

        inv.invokeFunction("hello", "Federico" );
    }
}
```

Per richiamare una funzione su uno script dobbiamo essere sicuri che l'interfaccia Invocable sia implementata dall'engine che stiamo utilizzando. Nella distribuzione ufficiale di Java 6 è possibile trovare solo l'engine per JavaScript, però basta collegarsi al sito https://scripting.dev.java.net/ per avere informazioni su quali altri engine che rispettano le specifiche della JSR233 e che quindi possiamo caricare nel nostro programma.

JAVA COMPILER API

Un'altra feature interessante introdotta con la versione 6 riguarda le Java Compiler API (JSR199). Queste API permettono di giocare a tempo di esecuzione con il compilatore Java e forniscono numerose informazioni per lo sviluppatore. Vediamo come poter utilizzare queste nuove classi che ci vengono fornite

```
import javax.tools;
import java.net.*;
import java.io.*;

public class TestCompilerApi {
        public static void main(String[] args)
                          throws Exception {
                    try {
                               JavaCompiler
jc = ToolProvider.getSystemJavaCompiler();
StandardJavaFileManager sjfm =
           jc.getStandardFileManager(null, null, null);
new File("c:\\Test.java");
fileObjects = sjfm.getJavaFileObjects(javaFile);
jc.getTask(null, sjfm, null, null, null,
                               fileObjects).call();
                               sjfm.close();
                    }
                    catch (Exception e) {
System.out.println(e.toString());
                    }
        }
}
```

Prima di tutto dobbiamo importare il package javax.tools, dove troviamo le nuove API.
A questo punto possiamo istanziare il compilatore Java, o meglio la classe che ci permette di gestirlo, ovvero JavaCompiler. Oltre ad un istanza di JavaCompiler abbiamo ottenuto anche uno StandardJavaFileManager, un'interfaccia che ci permette di gestire i file come oggetti. A questo punto dobbiamo caricare la classe che vogliamo compilare. Prima apriamo un file con questa classe e poi creiamo un oggetto a partire da questo file utilizzando StandardJavaFileManager. A questo punto possiamo tranquillamente far partire il task di compilazione. Se tutto è andato bene troveremo nella directory indicata la classe compilata che potremo richiamare tranquillamente nei nostri programmi.

DERBY, IL DATABASE EMBEDDED

Come sempre nelle nostre applicazioni, passate, presenti e future ci sarà sempre bisogno di un database. Ora all'interno dell'ultima distribuzione Java troviamo Derby, un database opensource firmato Apache.
Questo non è chiaramente l'invenzione del nuovo millennio, visto che già esistono diversi database embedded che possono essere utilizzati in Java. Questo però ha la fortuna di essere sponsorizzato dal gruppo Apache e di essere stato preso dalla Sun come database embedded per il suo linguaggio. Derby, o Java DB come è stato rinominato da SUN, ha tutte le caratteristiche che un database al giorno d'oggi deve avere, transazionale, sicuro, facile da usare, supporto JDBC, SQL e il tutto rimane in 2 mega, il che garantisce anche buone prestazioni. Qui di seguito vediamo un'applicazione che si collega a questo database embedded e lo utilizza come un qualsiasi db.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;

public class TestDerby {
```

```
public static String driver =
    "org.apache.derby.jdbc.EmbeddedDriver";
public static String protocol =
    "jdbc:derby:";
public static String username="username";
public static String password="password";
public static Connection conn = null;

public static void utilizzo() {
System.out.printl("Utilizzo: java TestDerby
    <action>");
System.out.printl("<action>=insert, select");
    System.exit(0);
}

public static void main(String[] args) {
    connessione();
    insert();
    select();
}

public static void connessione() {
    try {
System.out.printl("Connessione a Derby...");
Class.forName(driver).newInstance();
Properties props = new Properties();
props.put("user", username);
props.put("password",password);
conn =
    DriverManager.getConnection(protocol
    +"derbyDB;create=true", props);

System.out.println("Database creato e
    connesso");

conn.setAutoCommit(true);
    }
    catch(Exception e) {
System.out.println("Errore durante la connessione");
System.out.println(e.toString());
System.exit(0);
    }
}
public static void insert() {
    Statement s =
    conn.createStatement();
    s.execute("create table
    testDerby(num int, value int)");
s.execute("insert into testDerby values
    (2828,3828)");
    s.execute("insert into
    testDerby values (1234,5678)");
System.out.println("Insert effettuata");
}

public static void select() {
    ResultSet rs =
    s.executeQuery("SELECT * FROM testDerby");
```

```
    while (rs.next()) {
System.out.println("ID "+rs.getInt(1));
System.out.println("Value "+rs.getInt(2));
    }
System.out.println("Select effettuata");
    }
}
```

Come potete vedere dal programma d'esempio qui riportato, la connessione verso il database Derby è uguale a qualsiasi altro. C'è una sola cosa da notare rispetto ai classici database, nella prima connessione che viene fatta il database viene creato utilizzando la stringa aggiuntiva create=true nella stringa di connessione.

XML API

La JSR 173 è quella che definisce una nuova API, StAX (Streaming API for XML), che permette una diversa maniera per il parsing dei file XML. Nelle versioni precedenti di Java potevamo utilizzare JAXP 1.3 (Java API for XML Processing), che implementava le interfacce SAX per gli eventi (Simple API for XML) e DOM (Document Object Model) per la costruzione dell'albero di. In Java 6 è inclusa la versione 1.4 di JAXP che implementa l'interfaccia StAX. Questa nuova metodologia cerca di fornire al programmatore una semplice API, che al contrario delle vecchie API dà anche la possibilità di scrivere, oltre che leggere, informazioni XML. La metodologia di StAX rispetto a SAX può essere definita "pull". Praticamente in questo caso non è il parser che invia dati all'applicazione man mano che trova dati XML, ma è l'applicazione che richiede i dati di volta in volta. Java 6 quindi include un'implementazione completa di questa nuova metodologia, supportando la validazione e la trasformazione di documenti utilizzando questa API. Di seguito sono riportate tutte le API che sono comprese nella nuova versione di JAXP

- JAXP 1.4 specification
- SAX 2.0.2
- StAX 1.0, JSR 173
- XML 1.0, XML 1.1
- XInclude 1.0
- DOM Level 3 Core, DOM Level 3 Load and Save
- W3C XML Schema 1.0
- XSLT 1.0
- XPath 1.0

Rispetto alle precedenti metodologie di parsing ci sono chiaramente aspetti positivi e negativi, che dovrebbero essere valutati in fase di progetto per capire quale sia quello che si addice meglio al nostro caso.

NOTE

JAXP e JStAX sul Web

http://www.developer.com/xml/article.php/3397691
https://jaxp.dev.java.net/1.4/
http://java.sun.com/webservices/docs/1.6/tutorial/doc/SJSXP2.html
http://edocs.bea.com/wls/docs90/xml/stax.html

INNOVAZIONE DESKTOP

Per concludere la nostra panoramica su Java 6 non possiamo non parlare delle innovazioni che sono state introdotte dal punto di vista Desktop. In questa major release del linguaggio abbiamo per la prima volta la possibilità di non legare le nostre applicazioni a librerie JNI che accedono al sottostante sistema operativo, potendo utilizzare API che mettono già a disposizione qualcosa che prima era impensabile. Prima novità fra tutte la SystemTray. La maggiorparte delle applicazioni sui sistemi operativi moderni permettono di essere inserite nella barra di laterale, con una semplice icona. Fino alle versioni precedenti i programmi Java non potevano assolutamente accedere a questa barra, se non utilizzando progetti opensouce come systray4j o JDIC. In questa nuova versione è stata invece definita un API, multipiattaforma, che ci permette di gestire la System Tray in maniera anche abbastanza semplice. Questa barra viene gestita come un semplice oggetto grafico, al quale aggiungere un classico PopupMenu per gestire l'applicazione.

```
//Prima di tutto controlliamo se il sistema supporta l'API
if (!SystemTray.isSupported())
        System.exit(0);
JFrame frame;
SystemTray systemTray;
//Istanziamo la System Tray di sistema per poterla utilizzae.
systemTray=SystemTray.getSystemTray();
Image image=Toolkit.getDefaultToolkit().getImage("tray.gif");
//Ora creiamo il nostro menu da aggiungere alla barra
PopupMenu popupMenu;
popupMenu=new PopupMenu();
MenuItem item1=new MenuItem("Uno");
MenuItem item2=new MenuItem("Due");
MenuItem item3=new MenuItem("Tre");
popupMenu.add(item1);
popupMenu.add(item2);
popupMenu.add(item3);
//Infine aggiungiamo il menu alla barra
TrayIcon trayIcon=new TrayIcon(image,"Prova Tray",popupMenu);
tray.add(trayIcon);
```

Per quanto riguarda altre innovazioni che ha introdotto la nuova versione di Java ci sono sicuramente miglioramenti e nuove API. Quello che però più fa capire l'avvicinamento di questa versione al mondo Desktop è la classe java.awt.Desktop. Questa classe, che deriva direttamente dal progetto opensource JDIC (JDesktop Integration Components), permette principalmente allo sviluppatore di

- Lanciare il browser di default specificando un uri
- Lanciare il client mail di default specificando un indirizzo email
- Di far partire le applicazioni associate ad un certo tipo di file

Queste possibilità nelle precedenti versioni di Java erano impensabili e fanno intuire che le funzionalità di Java relativamente al mondo Desktop potrebbero non fermarsi qui, ampliando magari un API multipiattaforma che permetta un collegamento con il sistema operativo sottostante. Qui di seguito vediamo uno stralcio di codice che utilizza in sequenza le nuove funzionalità della classe Desktop

```
import java.io.*;
import java.awt.*;
import java.net.*;
…..
…..
…..
if(!Desktop.isDesktopSupported())
        System.exit(0);

Desktop desktop;
desktop=Desktop.getDesktop();

//Apriamo/modifichiamo/stampiamo il file con il viewer associato per default
File f=new File("test.doc");
desktop.open(f);
desktop.edit(f);
desktop.print(f);
…..
…..
…..
//Lanciamo il client di posta con una nuova email, prima vuota poi specificando l'email
desktop.mail();
desktop.mail(new URI("mailto:test@server.com"));
```

CONCLUSIONI

Con tutte le novità che abbiamo visto in questo articolo possiamo renderci conto di quanto questa nuova versione di Java abbia portato una ventata di freschezza. C'è da dire che ultimamente Java era già stato leggermente rivoluzionato dalla versione 5.0 Tiger, quindi per il classico programmatore che non ha molto tempo per dedicarsi agli aggiornamenti forse è arrivato il momento di seguire la strada che sta percorrendo Java con queste nuove versioni, per non perdere il treno che sta passando in questo momento ed utilizzare tutte le interessanti novità che pian piano stanno rivoluzionando il nostro linguaggio.

Federico Paparoni

L'autore, Federico Paparoni, può essere contattato per suggerimenti o delucidazioni all'indirizzo email f.paparoni@ioprogrammo.it

SVILUPPO JSF VISUALE CON NETBEANS

PANORAMICA SUL VISUAL WEB PACK DI NETBEANS: IL NUOVO ADD-ON PER LA CREAZIONE DI APPLICAZIONI WEB, BASATE SU JAVA SERVER FACES, IN MANIERA COMPLETAMENTE VISUALE. REALIZZIAMO VELOCEMENTE UN CATALOGO CONSULTABILE ONLINE

Oggigiorno una grande parte dello sviluppo del software è rappresentato dalle applicazioni web-based. Non a caso, la piattaforma Java che sta riscontrando maggior successo è quella enterprise. In passato, lo sviluppo di applicazioni su infrastrutture enterprise richiedeva notevoli sforzi e conoscenze di livello avanzato. Con l'avvento della piattaforma Java Enterprise, il grado di difficoltà è notevolmente diminuito. Le conseguenze indirette di tale introduzione sono la riduzione delle complessità di sviluppo e dei relativi costi. Questa tendenza, atta a semplificare questo tipo di applicazioni, è stata rafforzata dall'ultimo rilascio della piattaforma, denominata Java Enterprise Edition 5. L'ampio elenco di modifiche e novità introdotte (ad iniziare dal nome in cui non compare più il numero 2) è stato finalizzato dalle seguenti linee guida: facilità di sviluppo, aumento della produttività e coinvolgimento delle community di sviluppatori.
Sebbene i cambiamenti apportati abbiano interessato l'intera piattaforma, in questo articolo ci soffermeremo solo sulla parte relativa allo strato di presentazione, rappresentato dai moduli per la definizione dell'interfaccia utente. Sin dalle origini, le fondamenta di tale strato sono state basate sulla tecnologia delle Java Server Pages. Attraverso questa tecnologia è però difficile per gli sviluppatori creare interfacce grafiche accattivanti, di alta qualità e riutilizzabili, come, ad esempio, quelle basate su JFC/Swing. Per agevolarci in questo compito la piattaforma enterprise ci viene incontro con lo standard Java Server Faces: una tecnologia server-side e page-driven, basata su componenti, per la creazione di interfacce utente. Lo sviluppo di applicazioni basate su JSF presenta una struttura simile a quelle Swing, infatti, anche qui ci troveremo a lavorare con entità come eventi, azioni e soprattutto componenti riutilizzabili. Questa particolare struttura, oltre a facilitare il lavoro dello sviluppatore, è stata principalmente ideata per agevolare i vari vendor nella realizzazione di tool di per lo sviluppo grafico. In questo articolo non ci focalizzeremo sulla tecnologia JSF, ma illustreremo come con l'aiuto del Netbeans Visual Web Pack sia possibile creare, in maniera quasi del tutto visuale, un'applicazione web pur avendo una minima conoscenza di JSF.

COME INIZIARE

COME INIZIARE

1. È indispensabile avere installato sul computer Java 2 Standard Edition 1.5 Update 9 o superiore. È preferibile utilizzare l'ultima versione disponibile dal sito http://java.sun.com.
2. Scaricare il pacchetto Netbeans IDE 5.5 (circa 85 MB) contenente il Java EE Application Server 9.0 U1 (alias GlassFish) ed il Visual Web Pack (circa 45MB) dalla sezione Downloads del sito http://www.netbeans.org. Installare prima l'ambiente di sviluppo e successivamente il Visual Web Pack. Siamo pronti a sviluppare le nostre web applications!

Fig. 1: *La form di login della nostra applicazione nella vista Design*

L'APPLICAZIONE DI ESEMPIO

In questa prima parte dell'articolo ci siamo soffermati sugli aspetti teorici del tema in questione, in modo da dare, anche al lettore che si avvicina per la prima volta a questo tipo di tecnologia, un quadro d'insieme dell'ambiente in cui fra poco ci troveremo a lavorare. D'ora in avanti la nostra attenzione sarà interamente incentrata all'implementazione di una semplice applicazione web per la consultazione delle informazioni relative ai libri di testo contenuti in una biblioteca. L'applicazione presenterà una form di login in cui l'utente dovrà inserire lo username e la password di accesso al sistema, il quale, una volta verificata la correttezza delle informazioni inserite, visualizzerà un nuova pagina contenente le informazioni sui libri in forma tabellare. I dati di accesso dovranno essere obbligatori e nel caso in cui siano errati si dovrà visualizzare un opportuno messaggio e la possibilità di tornare alla pagina iniziale. Ovviamente, per l'interazione con i dati relativi ai libri ci serviremo di un database relazionale, la cui creazione è descritta nell'apposito riquadro.

DISEGNO DELL'APPLICAZIONE

La prima azione da fare è la creazione di un nuovo Visual Web Project dal menu *File -> New Project*. Lo scheletro del progetto verrà automaticamente generato dall'ambiente di sviluppo.
La realizzazione di ogni singola pagina con Java Server Faces necessita di due step: la creazione di una pagina JSP contenente i tag JSF per la definizione dei componenti ed una classe Java, denominata "*backing bean*", per la gestione logica dei componenti definiti nella pagina stessa. L'implementazione di una nuova pagina con Netbeans è delegata a tre viste differenti: Design, JSP e Java. Le ultime due danno accesso alla pagina JSP ed al backing bean, mentre la vista Design consente allo sviluppatore di aggiungere, via drag and drop, i componenti grafici, presenti nella Palette, per la definizione dell'interfaccia utente. Questo processo presenta molti punti in comune con la creazione di interfacce Swing attraverso il Matisse GUI Builder.
In Figura 1 *illustra* la struttura della pagina di login, creata in modo completamente visuale, comprendente una label di benvenuto, una *textfield* ed una *passwordfield* per l'inserimento dei dati di accesso ed un *button* per la richiesta di accesso ai dati della biblioteca.
Un componente grafico, una volta inserito, presenta delle caratteristiche e dei comportamenti di default. Per "modellarlo" secondo le proprie esigenze è necessario selezionarlo e modificarne le proprietà, visualizzate nella finestra *Properties* posta sulla destra. La maggior parte dei componenti presentano preziose funzionalità pensate per diminuire il lavoro degli sviluppatori, soprattutto per quanto riguarda la scrittura di codice di scripting necessario alla gestione dei componenti lato client. Ad esempio, nella pagina appena creata, i campi di testo per l'inserimento dei dati di accesso sono stati impostati come required. Attraverso questa semplice modifica, oltre all'aggiunta del carattere * (wildcard) che ne specifica l'obbligatorietà, sulla pressione del bottone di login verrà eseguito un controllo lato client che verificherà l'effettiva presenza dei dati. È inoltre importante sottolineare la facilità di posizionamento dei componenti, la cui modifica all'interno della griglia può variare con una precisione pari ad un pixel, se si tiene premuto il tasto Shift.
Terminata la parte grafica, non ci rimane che passare all'elaborazione dei dati inseriti dall'utente ed alla verifica dei permessi di accesso. Tale attività è fattibile cliccando due volte sul bottone di login: l'IDE ci porterà all'interno del metodo **loginButton_action** del backing bean, che implementeremo nel seguente modo:

```
public String loginButton_action() {
    String response = null;
if(this.usernameField.getText().equals
                                        ("ioProgrammo") &&
this.passwordField.getText().equals
                                        ("ioProgrammo")) {
        response = "loginOK";
```

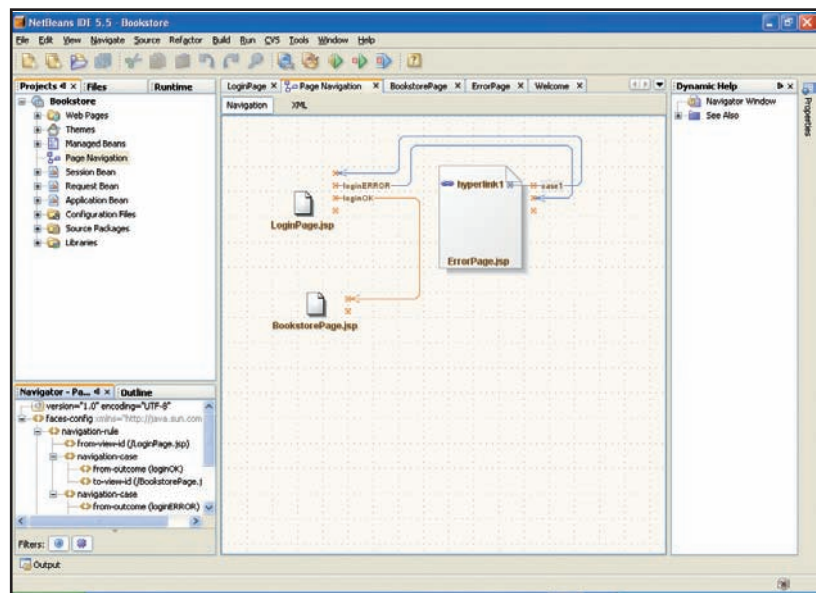



***Fig. 2:** Il flusso della nostra applicazione nella vista Page Navigation*

```
    } else {
        response = "loginERROR";
    }
    return response;
}
```

L'oggetto ritornato è di tipo String e può assumere i valori **loginOK** e **loginERROR** in base alla validità o meno dei dati inseriti. Come vedremo nel prossimo paragrafo, queste due stringhe saranno fondamentali nella costruzione del flusso dell'applicazione. Vista la natura dell'applicazione, l'implementazione dell'azione di login è stata fatta con una banale comparazione tra stringhe; in uno scenario diverso il metodo appena riportato potrebbe essere il punto di accesso ad un servizio di backend.

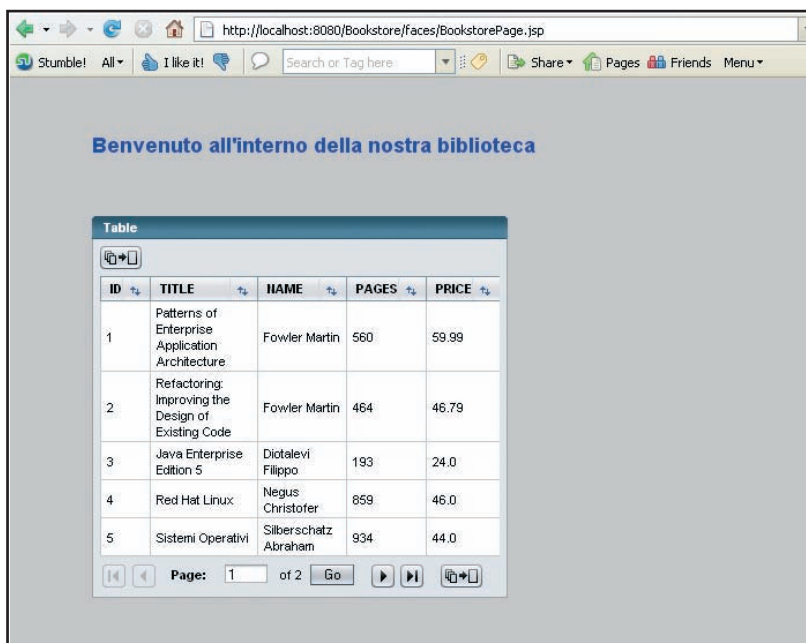



Fig. 3: Visualizzazione dei dati all'interno di un web-browser

SCELTA DELL'APPLICATION SERVER

Tutti i programmatori che si occupano di sviluppo di applicazioni enterprise basano il proprio lavoro su sistemi chiamati application servers. Tali sistemi, rappresentano l'implementazione delle specifiche redatte dagli ingegneri di Sun Microsytems. Attualmente, sul mercato sono presenti una miriade di application server, sia commerciali che open source, che presentano caratteristiche differenti sia in termini prestazionali che funzionali (es: supporto per il clustering). In questo articolo, per il deploy dell'applicazione di esempio, è stato utilizzato il prodotto open source Java Enterprise Edition Application Server 9.0 U1, meglio conosciuto con il nome di GlassFish, che rappresenta l'implementazione di riferimento delle specifiche Java EE 5 (https://glassfish.dev.java.net/). Inoltre, al suo interno è stato integrato il Java DB, un database relazionale open source, interamente implementato in Java, e derivato dal progetto Apache Derby (http://developers.sun.com/prodtech/javadb/index.jsp).

FLUSSO DELL'APPLICAZIONE

In Figura 2 è riportata la vista rappresentante il flusso dell'applicazione di esempio ottenibile cliccando sul nodo *Page Navigation* all'interno della struttura del progetto. Gli appassionati di mobile che sviluppano applicazioni basate sul profilo MIDP utilizzando il Mobility Pack di Netbeans noteranno immediatamente una forte somiglianza con il Visual Mobile Designer: infatti, entrambi gli strumenti sono stati costruiti utilizzando le stesse librerie.
Come accade nella definizione del flusso delle MIDlet, attraverso questo tool è possibile specificare i criteri di navigazione tra le diverse pagine. Nel caso in cui l'azione della pagina di login ritorni la stringa loginOK verrà richiamata la pagina BookstorePage.jsp. Se, invece, l'azione ritorna la stringa loginERROR il flusso verrà rediretto verso la pagina ErrorPage.jsp. In quest'ultima pagina è stato aggiunto un collegamento che riporta alla pagina iniziale. Sottolineiamo che anche questa attività è stata fatta in modo interamente visuale così come avverrà per la parte relativa all'integrazione e visualizzazione dei dati presenti sul database.

INTEGRAZION E DEI DATI

Eccoci arrivati al cuore della nostra applicazione: l'estrapolazione e la rappresentazione dei dati. Solitamente, questo compito richiede una serie di passi da seguire in cui è necessario avere conoscenze di sviluppo sia lato server (lettura dei dati da database) che lato client (presentazione dei risultati). Il Visual Web Pack ci svincola da queste problematiche in modo veloce è intuitivo.
Come si può vedere dalla Figura 3, la pagina BookstorePage.jsp è stata composta da una label ed una table. Per "collegare" i dati a quest'ultimo componente è sufficiente trascinare, partendo dalla finestra Runtime, il nodo che identifica la tabella BOOKS all'interno del componente stesso. In background, l'ambiente di sviluppo tradurrà questa nostra azione con la creazione di un session bean, contenente i comandi SQL da eseguire per la lettura dei dati, che sarà richiamato dal backing bean per la gestione della pagina in questione. Inoltre, cliccando con il tasto destro del mouse sulla tabella, attraverso la funzione Bind to Data …, ci verrà presentato un dialog che ci consentirà di scegliere quali

colonne del database visualizzare all'interno del componente grafico. Infine, selezionando la funzione Table Layout …, il tool ci darà la possibilità di definire le informazioni relative alla visualizzazione come, ad esempio, i titoli delle colonne, l'eventuale descrizione e la paginazione. In Figura 4 è riportato uno screenshot dell'applicazione eseguita all'interno di un web-browser.

CONCLUSIONI

Il Netbeans IDE rappresenta uno degli strumenti di sviluppo tra i più diffusi nelle community di sviluppatori. Tra i punti di forza ci sono sicuramente i cosiddetti tool WYSIWYG (What You See Is What You Get – quello che vedi è quello che ottieni), ad iniziare dal Matisse GUI Builder per interfacce Swing, per finire con il Visual Mobile Designer utile alla creazione di MIDlet. In questo articolo abbiamo fatto una panoramica sull'ultimo nato: il Visual Web Pack. La tecnologia Java Server Faces insieme a questo tool rivoluzionano il modo di sviluppare le interfacce grafiche per applicazioni Java su piattaforma enterprise. Dato che lo strumento è in versione Technology Preview vale la pena rimanere sintonizzati sulle successive evoluzioni del prodotto in modo da carpirne immediatamente le interessanti novità.

In ogni caso il backend composto dalle pagine JSF sicuramente rappresenta un'innovazione tecnologica di tutto rispetto per quanto riguarda Java

Fabrizio Fortino

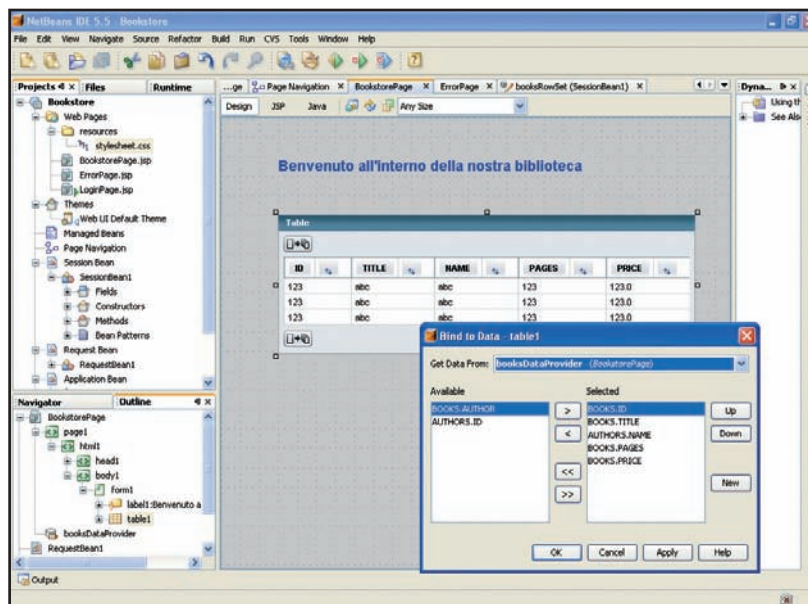
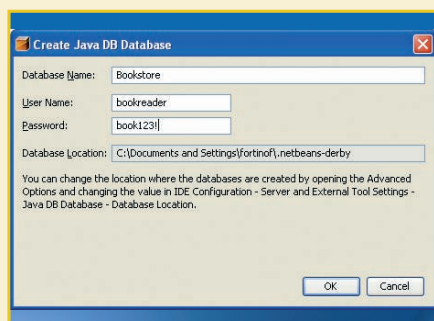



Fig. 4: La facility per il collegamento dei dati presenti sul database

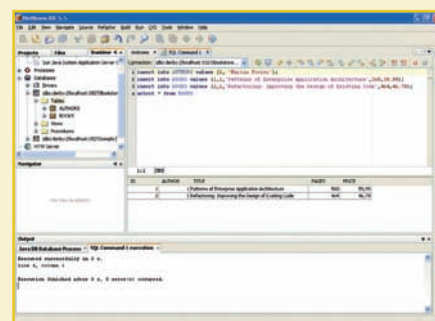
CREAZIONE DEL DATABASE

Per completare l'implementazione dell'applicazione di esempio è necessario disporre di una base dati contenente le informazioni sui libri di testo. In questo paragrafo illustreremo come crearlo, in maniera facile e veloce, utilizzando il Java DB integrato nell'application server e l'intuitivo wizard messo a disposizione dal Netbeans IDE

> CREAZIONE DI UN DB

1 Possiamo utilizzare il Wizard messo a disposizione dall'ambiente per la creazione del database. Selezioniamo il menu Tools -> Java DB Database -> Create Java DB Database. Adesso ci sarà richiesto di introdurre le informazioni base per l'inizializzazione.

> LE TABELLE

2 Selezioniamo la funzione Create Table…, cliccando con il tasto destro del mouse sul nodo Tables presente nella finestra Runtime. L'IDE ci presenterà un'agevole wizard con cui creeremo due tabelle: la prima per i libri (Books) e l'altra per gli autori (Authors)

> INSERIMENTO DEI DATI

3 A questo punto non ci rimane che popolare le due tabelle precedentemente create utilizzando la sintassi SQL ANSI 92. Chi non volesse utilizzare il wizard può skippare gli ultimi due step ed importare le tabelle, contenute nei files .grab, con il comando Recreate Table….

UN NEWS TICKER IN STILE WEB 2.0

IN QUESTO ARTICOLO IMPARERETE AD UTILIZZARE JAVASCRIPT PER SCRIVERE CODICE ORIENTATO AGLI OGGETTI. A TAL PROPOSITO VEDREMO COME IMPLEMENTARE L'OBSERVER PATTERN COSTRUENDO UN NEWS TICKER CHE RECUPERA LE NOTIZIE DAL WEB

Negli ultimi tempi siamo stati travolti dalla moda di Ajax. Ormai quasi tutti sanno che dietro questo termine si nasconde l'acronimo Asynchronous JavaScript and XML. È evidente, quindi, che il linguaggio di programmazione principalmente coinvolto, quando si parla di Ajax, è JavaScript. Fino a qualche tempo fa, questo potente linguaggio veniva utilizzato solo per compiti di minore rilievo. Con l'avvento di Ajax, tuttavia, JavaScript ha preso il volo e non ci si può più limitare a conoscerlo superficialmente. In questo articolo vedremo un po' di programmazione avanzata in JavaScript scrivendo del codice orientato agli oggetti ed implementando uno dei pattern fondamentali: l'Observer Pattern. Utilizzeremo tutti questi ingredienti per sviluppare un news ticker ampiamente personalizzabile come quello mostrato in figura 1.

Fig. 1: News Ticker in azione.

Ovviamente le news saranno recuperate utilizzando Ajax. Iniziamo subito col vedere come è possibile definire una classe in JavaScript.

DEFINIZIONE DI UNA CLASSE IN JAVASCRIPT

Chi si avvicina per la prima volta a JavaScript, provenendo da linguaggi object-oriented come Java e C#, si aspetta che per definire una classe sia presente un costrutto *class* o qualcosa di simile. Niente del genere, in JavaScript per definire una classe si usa la parola *function*. Ecco un esempio:

```
//dichiarazione della classe Persona
function Persona(nome, cognome)
{
        this.nome = nome;
        this.cognome = cognome;
}

//crea un'istanza della classe
var persona = new Persona("Alessandro", "Lacava");
//visualizza il valore della proprietà nome
alert(persona.nome);
```

Nel codice precedente abbiamo definito la classe Persona ed il suo costruttore il quale prende due parametri. Dopodiché si è creata un'istanza della classe. Chiamando *alert* viene poi visualizzata la proprietà *nome*. Vediamo, ora, come dichiarare un metodo della classe precedente:

```
Persona.prototype.visualizzaCognome = function()
{
        alert(this.cognome);
};
```

Tale codice aggiunge, alla classe *Persona*, il metodo *visualizzaCognome* il quale visualizza il valore della proprietà *cognome*. Il metodo appena dichiarato non prende alcun parametro. Ecco invece un esem-

pio di metodo che accetta dei parametri:

```
Persona.prototype.equals = function(persona)
{
        return (this.cognome == persona.cognome
                                                &&
        this.nome == persona.nome);
};
```

Il metodo *equals* restituisce true se l'oggetto passato come parametro ha le proprietà *cognome* e *nome* uguali all'oggetto in questione.

L'OBSERVER PATTERN

Detto in poche parole, un pattern è una soluzione comune e "rodata" ad un problema ricorrente. L'Observer Pattern è tra quelli di maggior uso. Ciò vuol dire che il problema risolto da tale pattern è ampiamente ricorrente durante lo sviluppo di software. La figura 2 illustra una versione semplificata del pattern in questione.

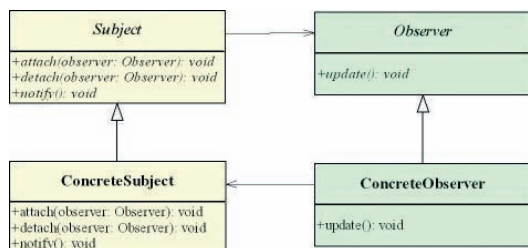



Fig. 2: ***Class Diagram semplificato dell'Observer Pattern.***

Per spiegare con una battuta l'essenza di questo pattern direi che, a grandi linee, corrisponde al vecchio detto: "L'Italia è il paese dove uno lavora e dieci guardano". Infatti, nell'Observer Pattern chi compie il lavoro è *Subject*, mentre quelli che "guardano" sono gli *Observer*. Utilizzando il metodo *attach* possiamo aggiungere gli observer al subject. Attraverso *detach*, invece, li rimuoviamo. Il metodo *notify* di *Subject* è utilizzato per notificare tutti gli observer dell'avvenuto completamento del lavoro. Sostanzialmente questa notifica avverrà richiamando il metodo *update* della classe *Observer*. Facciamo subito un esempio di applicazione di questo pattern analizzando il nostro news ticker.

IL NEWS TICKER

Un news ticker è un componente grafico che visualizza delle news, una dopo l'altra, dopo un determinato periodo di tempo. Un esempio è mostrato in figura 1. Ovviamente, l'immagine statica non rende l'idea del comportamento dinamico del ticker. Infatti, non è visibile il cambio tra una news e l'altra. Riferendoci al pattern di cui sopra, il nostro news ticker sarà composto da una classe, *Content Retriever*, che dovrà svolgere il lavoro di recuperare le news. Questa classe corrisponde, evidentemente, al *Subject* dell'Observer Pattern. La classe che invece rappresenta l'*Observer* è *Ticker*. Tale classe sarà notificata non appena il *ContentRetriever* recupererà le news. Il fetching delle news avverrà attraverso una chiamata Ajax ad uno script che gira sul server e che abbiamo chiamato *tickerData Retriever.php*. Da notare che siete liberi di implementare il lato server col linguaggio che preferite dato che la parte client prescinde da esso. La figura

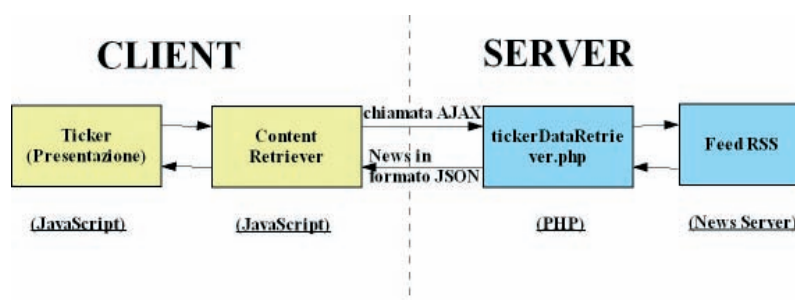



Fig. 2: ***Architettura del nostro ticker.***

3 dovrebbe chiarire i concetti esposti fino ad ora a parole. Quello che faremo, ora, è vedere come viene creato il nostro ticker all'interno di una pagina HTML e dopo analizzeremo tutto il "giro" in dettaglio. La prima cosa da fare è includere i file JavaScript necessari:

UN FLASH SU JSON

JSON sta per JavaScript Object Notation. In pratica è un formato di interscambio dati che utilizza la notazione letterale di JavaScript per rappresentare oggetti ed array. Sostanzialmente, quindi, è un sottoinsieme di JavaScript. Non entrerò nel dettaglio di JSON dato che gli è stato dedicato un intero articolo su ioProgrammo N. 108 – Novembre 2006. Un oggetto, in JSON, è rappresentato con la seguente sintassi:

```
{
"cognome" : "Lacava",
"nome" : "Alessandro",
"anni" : 30
}
```

Il precedente esempio rappresenta in modo letterale un oggetto Persona. Per rappresentare un array, invece, si usa la seguente sintassi:

```
{
  "linguaggi" : [
  "Java",
  "C#",
  "JavaScript"
  ]
}
```

Abbiamo così rappresentato l'array linguaggi. Per "compilare" un oggetto/array JSON si può utilizzare la funzione eval. Esistono anche diverse librerie lato server per mappare da linguaggi come PHP, Java e C# a JSON. Torniamo ora al nostro script lato server.

```
<script src="./scripts/ajax.js"
                type="text/javascript"></script>
<script src="./scripts/ContentRetriever.js"
                type="text/javascript"></script>
<script src="./scripts/Ticker.js"
                type="text/javascript"></script>
```

Il file *ajax.js* contiene il codice necessario alla creazione dell'oggetto *XMLHttpRequest* (XHR) tenendo conto delle differenze tra i browser in circolazione. Effettivamente contiene una sola funzione, *createHttpRequest*, che restituisce l'oggetto XHR. I file *ContentRetriever.js* e *Ticker.js* contengono, rispettivamente, le classi responsabili del recupero delle news e della successiva visualizzazione. Per ragioni di spazio non sarà analizzata la funzione *createHttpRequest* che peraltro è stata già vista in un mio precedente articolo su JSON (ioProgrammo N. 108 – Novembre 2006). La troverete in ogni modo nel codice allegato. Dopo aver incluso i file necessari, potete creare il news ticker utilizzando le seguenti righe di codice all'interno della vostra pagina HTML:

```
<div id="newsTicker"
        class="tickerBody">Caricamento news in
                corso...</div>
<script type="text/javascript" language="javascript">
    //5000 indica una rotazione della news dopo ogni
                5 secondi
        var ticker = new Ticker
                ("newsTicker", 5000, true);
        var contentRetriever = new
        ContentRetriever("./tickerDataRetriever.php");
        contentRetriever.attach(ticker);
        contentRetriever.getContent("http://punto-
                informatico.it/fader/pixml.xml");
</script>
```

In pratica, il precedente script crea gli oggetti di tipo *Ticker* e *ContentRetriever*. In seguito aggiunge il ticker (l'observer) al content retriever (il subject). Alla fine, chiama il metodo *getContent* di *ContentRetriever* passandogli l'URL del feed desiderato. Vediamo, ora, di analizzare le classi *Ticker* e *ContentRetriever*.

LA CLASSE TICKER

Quello che segue è il codice concernente il costruttore della classe *Ticker*:

```
function Ticker(elemId, delay, stopOnMouseOver)
{
        this.elemId = elemId;
        this.delay = delay;
        this.stopOnMouseOver = stopOnMouseOver;
        this.isMouseOver = false;
        this.dataArray = [];
        this.currentIndex = 0;
        this.errorMessage = "Nessuna notizia
                disponibile";
        var instance = this;
        if(this.stopOnMouseOver)
        {
document.getElementById(this.elemId).onmouseover
        = function(){instance.isMouseOver=true;};
document.getElementById(this.elemId).onmouseout =
        function(){instance.isMouseOver=false;};
        }
}
```

Il costruttore riceve tre parametri:

- *elemId*: l'id del blocco in cui visualizzare le news. Nel nostro caso il blocco sarà rappresentato dal seguente <div> che trovate nel codice HTML allegato:

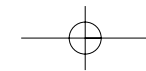
```
<div id="newsTicker"
        class="tickerBody">Caricamento news in
                corso...</div>
```

  Quindi elemId sarà costituito, in questo caso, da *newsTicker*.
- *delay*: il tempo, espresso in millisecondi, dopo il quale visualizzare la prossima news.
- *stopOnMouseOver*: un booleano che indica se fermare la rotazione delle news quando il mouse si posiziona sull'area relativa al ticker.

Le altre proprietà hanno il seguente significato:

- *isMouseOver*: un booleano che indica se il mouse è correntemente sopra il news ticker.
- *dataArray*: un array che contiene le news vere e proprie ricevute da *ContentRetriever*.
- *currentIndex*: un intero che rappresenta l'indice della notizia attualmente visualizzata.
- *errorMessage*: una stringa rappresentante il messaggio da visualizzare nel caso in cui le news non vengano recuperate correttamente.

Il restante codice aggiunge due event listener che gestiscono il bloccaggio delle news nel caso in cui il mouse si trovi sopra il ticker. Vediamo, ora, il metodo *update* richiamato da *ContentRetriever* non appena scarica le news. Il parametro passato in ingresso è proprio un array di news:

```
Ticker.prototype.update = function(dataArray)
{
        if(!dataArray || dataArray.length == 0)
        {   //si è verificato un errore.
                this.dataArray =
```

```
                                    [this.errorMessage];
        }
        else
        {   //nessun errore
                this.dataArray = dataArray;
        }
        this.rotateItems();
};
```

Il metodo *update* valorizza l'array di news col messaggio d'errore nel caso in cui il recupero delle notizie non va a buon fine. Se, invece, le news sono state scaricate senza problemi, valorizza l'array con le notizie vere e proprie. Infine, chiama il metodo *rotateItems* che si occupa di visualizzare le news, una dopo l'altra, ad intervalli di tempo costanti. Segue il codice di *rotateItems*:

```
Ticker.prototype.rotateItems=function()
{
        this.normalizeIndex();
        var instance = this;
        if(this.stopOnMouseOver &&
                                this.isMouseOver)
        {
                var POLL_TIME = 100;
                timer =
      setTimeout(function(){instance.rotateItems();},
                                        POLL_TIME);
        }
        else
        {
                var tickerElement =
            document.getElementById(this.elemId);
                var tickerContent =
             this.dataArray[this.currentIndex];
                tickerElement.innerHTML =
                                    tickerContent;
                timer =
  setTimeout(function(){instance.next();}, this.delay);
        }
};
```

Il precedente codice chiama il metodo *normalizeIndex* che resetta l'index a zero nel caso in cui esso abbia superato il numero di news presenti nell'array. Dopodichè, se il mouse è sopra il ticker, ne ferma la rotazione controllando l'uscita del mouse ogni *POLL_TIME* millisecondi. Dopo tale intervallo *rotateItems* richiama se stesso finché il mouse lascia l'area riguardante il ticker. Quando accade ciò viene inserita nel blocco, identificato da elemId, la news correntemente puntata da *currentIndex*. In seguito chiama il metodo *next* dopo *delay* millisecondi. Ecco il codice di *next*:

```
Ticker.prototype.next = function()
{
```

```
        clearTimeout(timer);
        this.currentIndex++;
        this.rotateItems();
};
```

Come si può vedere, esso non fa altro che resettare il timer, incrementare l'indice corrente e richiamare *rotateItems* che fa riprendere il ciclo. Chiaramente tutto ciò servirebbe a ben poco se non ci fosse qualcuno che facesse il lavoro di interrogare un server per recuperare le news. Come già detto, questo "sporco" lavoro spetta al nostro *ContentRetriever*.

LA CLASSE CONTENTRETRIEVER

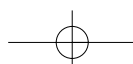
Il costruttore di ContentRetriever prende un solo parametro. Vediamone il codice:

```
function ContentRetriever(dataRetrieverUrl)
{
        this.dataRetrieverUrl = dataRetrieverUrl;
        this.jsonData = [];
        this.ajaxObj = createHttpRequest();
        this.observerArray = [];
}
```

L'unico parametro richiesto è l'URL del componente server responsabile di recuperare, fare il parsing e restituire le news in formato JSON. Nel nostro caso tale componente è costituito da *tickerDataRetriever.php* che vedremo più avanti. Le altre proprietà hanno il seguente significato:

- *jsonData*: l'array di news in formato JSON.
- *ajaxObj*: l'oggetto *XMLHttpRequest* recuperato in una maniera cross-browser attraverso la funzione *createHttpRequest* definita in *ajax.js*.
- *observerArray*: l'array di oggetti di tipo Observer. Nel nostro caso ne abbiamo solo uno, *Ticker*.

LA FUNZIONE EVAL

La funzione eval è molto potente ma altrettanto pericolosa. Essa esegue, infatti, qualsiasi codice JavaScript che le viene passato. Se non siete sicuri della fonte da cui proviene la stringa da parsare, conviene utilizzare delle librerie apposite che effettuano alcuni controlli di sicurezza ed eseguono il parsing solo se riconoscono il messaggio come JSON. Niente paura però, il parsing continua ad essere molto semplice (sempre una riga di codice!). Inoltre, esistono molte librerie per convertire un messaggio JSON in e dai principali linguaggi di programmazione quali: PHP, Java, C# e tanti altri. Una di quelle inerenti PHP l'abbiamo utilizzata proprio in questo articolo. Maggiori informazioni le potete trovare sulla pagina ufficiale di JSON: http://www.json.org/

Il metodo principale di *ContentRetriever* è *getContent*:

```
ContentRetriever.prototype.getContent =
                                    function(rssUrl)
{
        if (this.ajaxObj)
        {
                var instance = this;
                var parameters = "rssUrl=" +
                                              rssUrl;
                this.ajaxObj.open('GET',
   this.dataRetrieverUrl + "?" + parameters , true);
                this.ajaxObj.onreadystatechange =
            function(){instance.callbackFunction();};
                this.ajaxObj.send(null);
        }
};
```

Esso, attraverso una chiamata Ajax, richiama il nostro script PHP (contenuto nella proprietà *dataRetrieverUrl*) passando, per mezzo della query string, l'URL del feed da cui recuperare le news. Non appena le news saranno scaricate, verrà invocata la funzione di callback che ho chiamato, con molta fantasia, *callbackFunction*:

```
ContentRetriever.prototype.callbackFunction =
                                        function()
{
        if (this.ajaxObj.readyState == 4 &&
                            this.ajaxObj.status==200)
        {
                if(this.ajaxObj.responseText &&
                  this.ajaxObj.responseText.length > 0)
                {       //il response non è
                                              vuoto
                        try
                        {
                                this.jsonData
         = eval("(" + this.ajaxObj.responseText + ")");
                        }
                        catch(e)
                        {
                                this.jsonData
                                              = [];
                        }
}
                this.notify();
        }
};
```

Questa funzione controlla che la richiesta sia stata completata con successo. Questo si capisce dalle proprietà *readyState* e status dell'oggetto *XMLHttp Request* rappresentato da *ajaxObj*. Dopodichè, se il response non è vuoto, valorizza l'array *jsonData* con le news scaricate. Ricordiamo che eval è una funzione che interpreta "al volo" qualsiasi codice JavaScript (JSON incluso). Se il testo ricevuto non è JavaScript allora eval lancia un'eccezione che gestiamo utilizzando il blocco try-catch. Così facendo, in caso di errore, il ticker riceverà un array vuoto e visualizzerà il messaggio adatto. Da notare che se non avessimo utilizzato il blocco try-catch, in caso di errore, il contenuto del nostro <div> non sarebbe mai stato aggiornato. Avrebbe, quindi, continuato a visualizzare la stringa "Caricamento delle news in corso…".
Alla fine viene chiamato notify il quale notifica tutti gli observer dell'avvenuto recupero delle news:

```
ContentRetriever.prototype.notify = function()
{
        for(var i = 0; i < this.observerArray.length;
                                                i++)
        {
        this.observerArray[i].update(this.jsonData);
        }
};
```

Il metodo notify semplicemente chiama il metodo update di tutti gli observer che si sono registrati alla classe ContentRetriever attraverso il suo metodo attach. Questo metodo non fa altro che aggiungere un elemento all'array di oggetti observer:

```
ContentRetriever.prototype.attach =
                              function(observerObject)
{
        this.observerArray.push(observerObject);
};
```

Per ragioni di spazio non sarà esaminato il metodo detach il quale non fa altro che rimuovere un observer da ContentRetriever. Lo potete in ogni modo esaminare guardando il codice allegato.
Arrivati a questo punto manca un solo tassello per completare tutto il giro. Esso è costituito dal nostro script PHP che si occupa di recuperare, "parsare" e restituire le news in formato JSON.

IL COMPONENTE LATO SERVER

Le news che andremo a recuperare sono costituite da feed RSS. Per farne il parsing abbiamo utilizzato una libreria open source abbastanza nota. Essa è MagpieRSS. Inoltre, abbiamo usato un'altra libreria open source per trasformare le nostre news da array PHP ad array JSON. Questa libreria è JSON-PHP. In un box laterale potete trovare i link relativi ad entrambe le librerie. Ecco, invece, il codice concernente tickerDataRetriever.php:

```
require('magpierss/rss_fetch.inc');
require('json/JSON.php');
$DEFAULT_URL = "http://punto-
                    informatico.it/fader/pixml.xml";
$url = isset($_GET["rssUrl"]) ? $_GET["rssUrl"] :
                                        $DEFAULT_URL;
$rss = fetch_rss($url) or die('["Nessuna news
                                        disponibile"]');
echo contentAsJson($rss);
[...]
```

Le prime due righe di codice servono ad includere le librerie necessarie al parsing dei feed e alla conversione delle news in oggetti JSON. In seguito viene recuperato l'URL, passato attraverso la query string, che punta al feed vero e proprio. In caso non fosse passato, utilizziamo come feed di default quello di punto informatico. La funzione fetch_rss recupera ed esegue il parsing del feed RSS passato come parametro. La riga successiva trasforma, attraverso la funzione contentAsJson, le news in formato JSON e le restituisce nel responseText dell'oggetto XMLHttpRequest relativo a ContentRetriever. Ecco il codice di contentAsJson:

```
function contentAsJson($rss)
{
        $arr = array();
        $counter = 0;
        foreach ($rss->items as $item)
        {
                $title = $item["title"];
                $url   = $item["link"];
                $description =
                                $item["description"];
                $arr[$counter++] =
                        buildHtml($title, $url, $description);
        }
        $json = new Services_JSON();
        return $json->encode($arr);
}
```

Come si può vedere, vengono recuperati titolo, link e descrizione di ogni news. L'array $arr conterrà, alla fine del ciclo, tutte le news formattate attraverso la funzione buildHtml. Infine, è utilizzata la libreria PHP-JSON per trasformare l'array PHP in un array JSON. Notate che per fare ciò sono bastate due righe di codice.
Ecco, invece, il codice della funzione buildHtml:

```
function buildHtml($title, $url, $description)
{
        $html = "<a class='title' href='$url'
                                target='_blank'>".
$title.
"</a>".
"<br />".
```

```
                                                "<a
        class='description' href='$url' target='_blank'>".
$description.
"</a><br /><br />";
            return $html;
}
```

Essa non richiede particolari delucidazioni. Infatti, costruisce semplicemente l'HTML relativo ad ogni news e lo restituisce al chiamante. E' da notare che abbiamo utilizzato un foglio di stile esterno per permettere una completa customizzazione del formato da applicare alle news. Troverete il foglio di stile e tutto il resto nel codice allegato al presente numero della rivista.

CONCLUSIONI

Come abbiamo visto, JavaScript non è per niente un linguaggio da snobbare e/o usare solo per "roba di poco conto" come la validazione di un campo o la visualizzazione di una pop up. In questo articolo lo abbiamo utilizzato per costruire un news ticker ampiamente personalizzabile. Lo abbiamo fatto applicando una mentalità ad oggetti ed implementando l'Observer Pattern. Il vantaggio di ciò sta nel fatto che possiamo estendere la nostra applicazione in modo molto robusto. Ad esempio, sarebbe possibile implementare un altro observer basato sul comportamento dell'elemento <marquee>. Questo, invece di presentare le news in rotazione, le visualizza con un continuo scrolling. Una volta creata la classe JavaScript Marquee ci basterebbe istanziarla ed aggiungerla al ContentRetriever attraverso il suo metodo attach. Così facendo potremmo avere due ticker nella stessa pagina: uno che fa ruotare le news ed uno che le "scrolla". Ovviamente lo sviluppo della classe Marquee è lasciato per esercizio. Buon lavoro!

Alessandro Lacava

AVVIO DEL NEWS TICKER

**Per provare l'esempio di quest'articolo avrete bisogno di PHP e di un Web server.
Il primo lo potete trovare qui: http://www.php.net/
Come Web server io ho utilizzato Apache HTTP Server che potete trovare all'indirizzo: http://httpd.apache.org/
Se utilizzate Windows e volete un'installazione molto semplice sia di PHP sia del Web server vi consiglio di utilizzare EasyPHP. Esso installa, in un colpo solo, PHP, Apache Web server, MySQL e PHPMyAdmin. L'indirizzo per EasyPHP è il seguente: http://easyphp.org/
Una volta installato ed avviato il server, potrete avviare il ticker utilizzando un URL simile al seguente: http://localhost/ticker/ticker.htm
Questo supposto che avete impostato Apache HTTP Server in ascolto sulla porta di default (80) e messo tutto il codice allegato sotto la cartella ticker.**

MagpieRSS:
http://magpierss.sourceforge.net/
JSON-PHP:
http://mike.teczno.com/json.html
Sito ufficiale su JSON:
http://www.json.org/

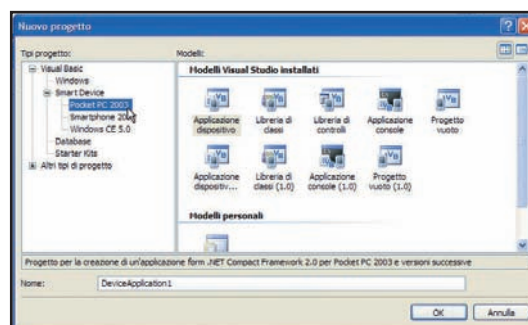
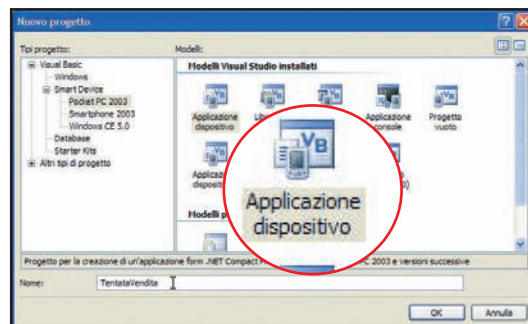
A SPASSO CON MS SQL SERVER

AVETE FORNITO AL VOSTRO COMMERCIALE UN PALMARE CON CUI PUÒ EFFETTUARE ORDINI E CONSULTARE IL DB AZIENDALE. SE NON VOLETE CHE RESTI PERENNEMENTE CONNESSO, DOVETE SINCRONIZZARE I DATI CON IL SERVER CENTRALE...

In passato, lo sviluppo di applicazioni per dispositivi mobili ha sempre rappresentato una sfida impegnativa per gli sviluppatori, data la sostanziale differenza tra questo tipo di applicazioni e le applicazioni desktop convenzionali. La situazione è ora radicalmente cambiata grazie anche a Visual Studio .NET 2005. I dispositivi mobili si stanno, ormai, avvicinando sempre di più a dei piccoli Personal Computer in miniatura. Come naturale conseguenza, ai dispositivi mobili si possono applicare, fatte alcune opportune considerazioni, i modelli di sviluppo esistenti finora, utilizzati per i desktop. In questo articolo analizzeremo il tipico problema che lo sviluppatore deve affrontare in una classica applicazione per palmari, come può essere la tentata vendita, in cui l'attore principale è l'agente di vendita che durante il suo giro dai clienti deve poter inserire gli ordini e poi deve sincronizzare i dati con la sede centrale

HELLO WORLD

In Visual Studio .NET, possiamo utilizzare gli stessi strumenti di progettazione, sviluppo, debug e distribuzione che utilizziamo per le convenzionali applicazioni desktop, anche per le applicazioni per dispositivi mobili.
Nella finestra di Progettazione Windows Form, VB mostra un emulatore di smart device in cui è possibile progettare rapidamente interfacce utenti, così come verranno visualizzate nel dispositivo reale. Allo stesso modo in cui si disegna l'interfaccia di una applicazione Windows form, si possono aggiungere controlli, impostare proprietà e scrivere codice per i gestori eventi con le stesse modalità utilizzate per le applicazioni classiche.
L'emulatore permette, inoltre, di verificare il corretto funzionamento di un'applicazione per Smart Device dal proprio PC, senza che sia collegato fisicamente un dispositivo fisico specifico per l'esecuzione.
Realizziamo il classico Hello World che si mette in pratica quando si inizia a sviluppare in un nuovo ambiente.

1 Per iniziare dobbiamo creare un nuovo progetto scegliendo la voce *Nuovo progetto* dal menu File. In questo modo viene visualizzata la finestra di dialogo *Nuovo progetto*, da questa finestra, nella visualizzazione ad albero *Tipi progetto*, dobbiamo espandere la voce *Visual Basic*, successivamente la voce *Smart Device*, e quindi cliccare su *Pocket PC 2003*.

2 Nel riquadro *Modelli* dobbiamo cliccare su *Applicazione dispositivo*, ed inserire il nome della nostra applicazione nella casella *Nome*, quindi scegliere *OK*, in questo modo nella finestra *Progettazione Windows Form* verrà visualizzata la rappresentazione di un dispositivo Pocket PC.

Per aggiungere controlli all'interfaccia, e per la scrittura del codice, si utilizza sempre il solito procedimento utilizzato per le applicazioni Windows Form classiche. Risulta subito evidente, che *form1* rappresenta un dispositivo mobile, ma oltre a ciò, la principale differenza, consiste nel numero inferiore di controlli di-

sponibili nella casella degli strumenti. Inseriamo il classico bottone in *form1* e scriviamo nel codice dell'evento *Click*:

```
Private Sub Button1_Click(ByVal sender As
            System.Object, ByVal e As System.EventArgs)
                                Handles Button1.Click
    MsgBox("Ciao, Mondo!")
End Sub
```

Eseguiamo l'applicazione e, dall'elenco degli emulatori disponibili, nella finestra di dialogo *Distribuisci*, selezioniamo la voce *Pocket PC 2003 SE - Emulatore*. Se disponiamo di un dispositivo fisico connesso al computer di sviluppo, possiamo eseguire direttamente l'applicazione su tale dispositivo.

Nella barra di stato viene visualizzato lo stato di avanzamento, nel momento in cui l'applicazione viene eseguita sull'emulatore, possiamo cliccare sul nostro bel pulsante per assicurarci che venga visualizzato il messaggio "Ciao, Mondo!".

SINCRONIZZAZIONE DEI DATI

Dopo aver intravisto come realizzare un'applicazione per dispositivi mobili, possiamo passare all'argomento centrale di questo articolo: la Sincronizzazione dei dati

SQL Server 2005 Mobile Edition permette di stabilire una connessione e scambiare dati con un database di SQL Server in due modi:

- Remote data access (RDA)
- Merge replication (replica di tipo merge)

La funzionalità RDA permette, ad un'applicazione VB, di accedere ai dati da una tabella di database remota di SQL Server, e di archiviarli in una tabella di database locale di SQL Server Mobile. Dopo che i dati sono stati archiviati nella tabella locale di SQL Server Mobile, l'applicazione potrà accedere alla tabella locale e quindi leggere ed aggiornare i dati in essa contenuti.

Una volta che l'applicazione termina di aggiornare i dati locali, può aggiornare nuovamente i record modificati dalla tabella locale nella tabella remota di SQL Server.

La propagazione dei dati dalla tabella remota di SQL Server ad una tabella locale di SQL Server Mobile viene definita come: *Pull* dei dati.

La propagazione delle modifiche apportate nella tabella locale di SQL Server Mobile alla tabella remota di SQL Server viene definita come: *Push* dei dati.

La funzionalità RDA, si può utilizzare quando non è necessario impiegare tutte le funzionalità messe a disposizione dalla replica di tipo merge come, ad esempio, la risoluzione di conflitti.

La replica di tipo merge, in generale, è la metodologia più adatta da utilizzare nei dispositivi mobili poiché permette di aggiornare i dati in modo autonomo e indipendente sul dispositivo portatile e sul server. Quando il dispositivo è connesso al server, è possibile sincronizzare i dati su entrambi, sia per inviare le modifiche dal client al server e sia per ricevere le nuove modifiche dal server.

Rispetto alla metodologia RDA, la replica di tipo merge richiede una configurazione e una manutenzione aggiuntiva sul server, ma in Sql Server 2005 la configurazione della replica è stata semplificata.

LA REPLICA DI TIPO MERGE

Analizziamo in dettaglio la replica di tipo merge descrivendo i due nuovi wizards messi a disposizione da SQL Server 2005 Management Studio

- Creazione del Publishers

● Creazione dei Subscribers

Il primo passo è, naturalmente, quello della creazione del database Sql Server 2005 che dovrà essere successivamente pubblicato. Il database Sql Server Mobile sarà invece sottoscrittore della pubblicazione Sql Server 2005. Creiamo subito un database dal nome TentaVendita con le seguenti tabelle ed i seguenti campi:

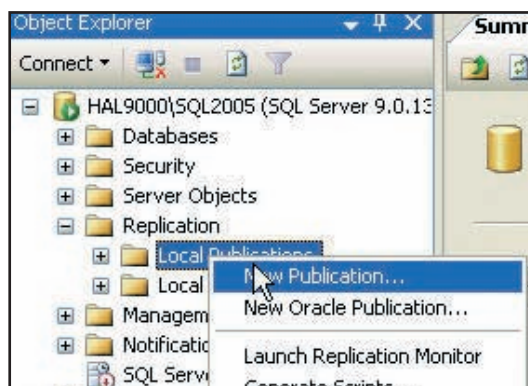
● tblArticoli
  ✓ IdArticolo,
  ✓ Descrizione ,
  ✓ Prezzo,
  ✓ Iva

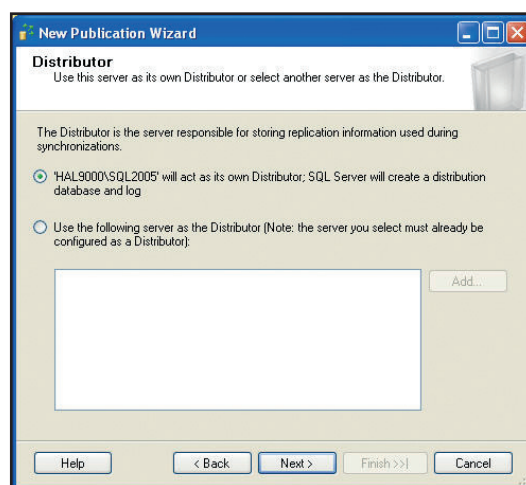
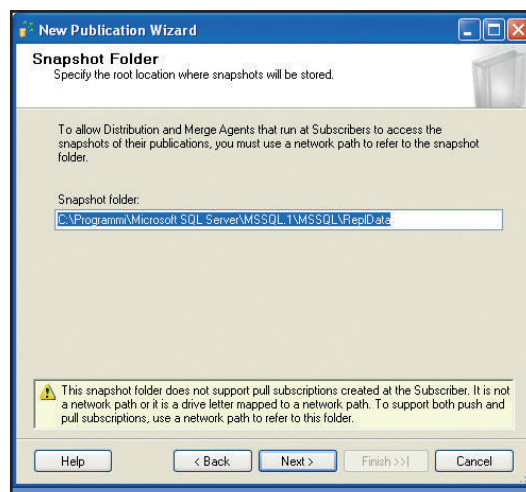
● tblClienti
  ✓ IdCliente
  ✓ Descrizione
  ✓ Indirizzo ,
  ✓ Citta ,
  ✓ Provincia

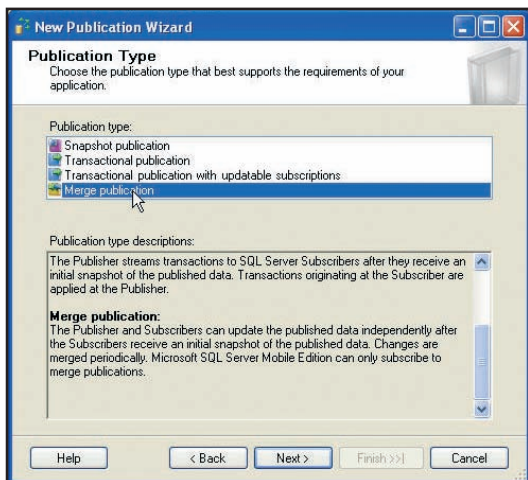
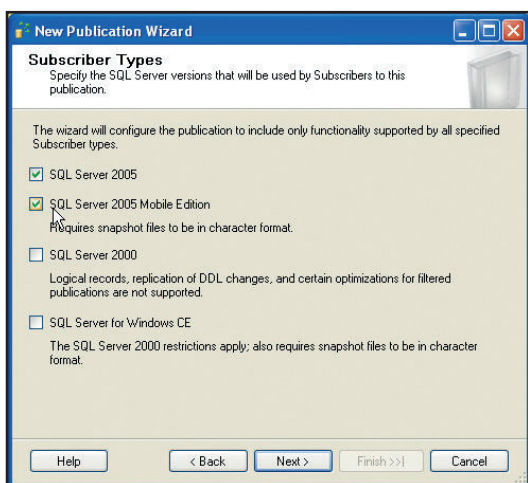
● tblOrdini
  ✓ IdOrdine,
  ✓ idCliente,
  ✓ Note

● tblRigaOrdine
  ✓ IdRiga,
  ✓ IdOrdine,
  ✓ IdArticolo,
  ✓ Prezzo,
  ✓ Iva,
  ✓ Quantita

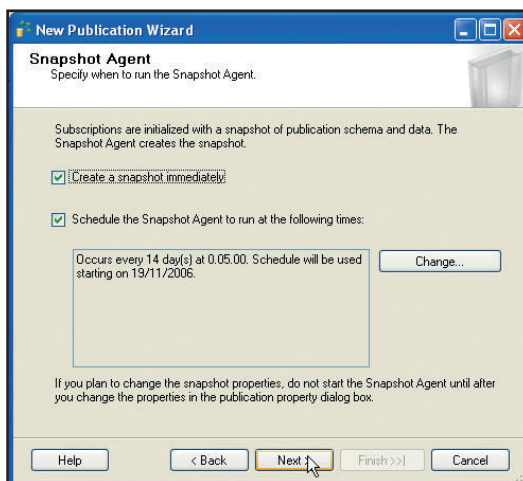
Dopo aver creato il database, ed averne impostato la struttura, possiamo creare una pubblicazione

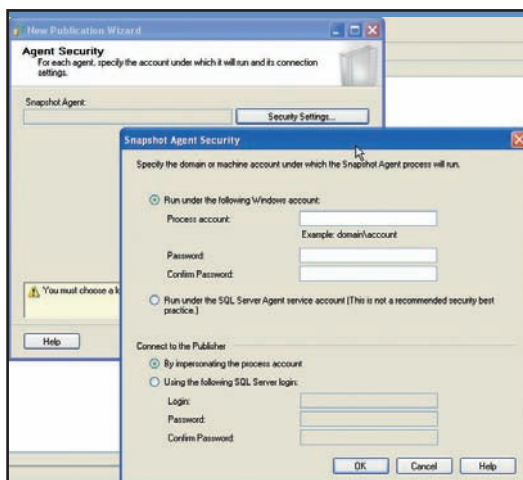
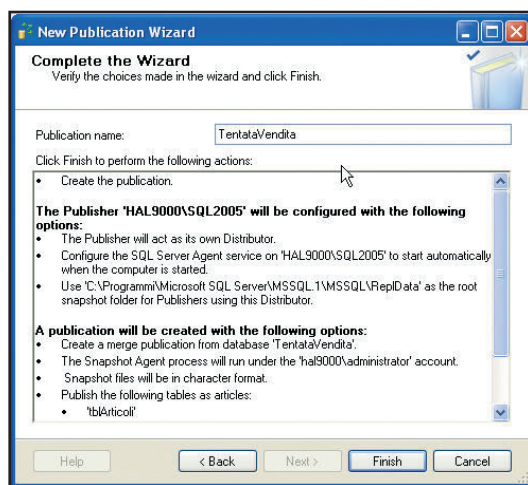
1 Nella finestra Object Explorer, espandiamo i vari nodi fino ad espandere il nodo Replication. A questo punto possiamo cliccare, con il pulsante destro del mouse, sulla

cartella *Local Pubblication* e selezionare, dall'elenco a discesa, la voce *New Publication*.

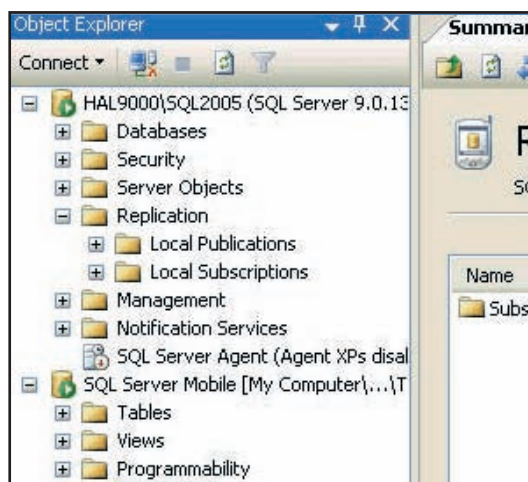
2 Nella schermata iniziale della Creazione guidata possiamo tranquillamente cliccare su Next. Viene, ora richiesto, di configurare il server di distribuzione, possiamo selezionare la prima opzione che ci permette di utilizzare il computer locale stesso come server di distribuzione, e quindi cliccare su Next.

3 Dopo aver scelto di far partire Sql Server Agent automaticamente all'avvio del sistema, ci verrà richiesto di indicare la cartella snapshot. Possiamo lasciare il percorso di default oppure possiamo immettere il percorso di condivisione di una cartella snapshot creata in precedenza. E' consigliabile immettere il percorso nella forma \\computer\CartellaSnapshot, dove computer indica il nome del computer in uso.

4 A questo punto verrà mostrato l'elenco dei database installati, da questo elenco ovviamente selezioniamo il database Ten-

tataVendita e clicchiamo su Next.
Nella successiva finestra viene mostrato l'elenco dei tipi di pubblicazione in cui dobbiamo selezionare la voce Merge Publication.

5 Nella pagina Subscriber types dobbiamo indicare le versioni di Sql da utilizzare, pertanto dobbiamo selezionare la casella accanto a SQL Server 2005 Mobile Edition per attivare il supporto per i Sottoscrittori di SQL Server Mobile.

6 Nell'elenco di oggetti da pubblicare, dobbiamo selezionare la casella di controllo Tables, in questo modo, se espandiamo il nodo Tables, possiamo notare come siano selezionate tutte e

quattro le tabelle del database SQL. A questo punto, clicccando su Next, verrà visualizzato un avviso, tale avviso ci informa che verranno aggiunte degli identificatori univoci alle tabelle, questo perchè tutte le tabelle di merge devono contenere una colonna uniqueindentifier.

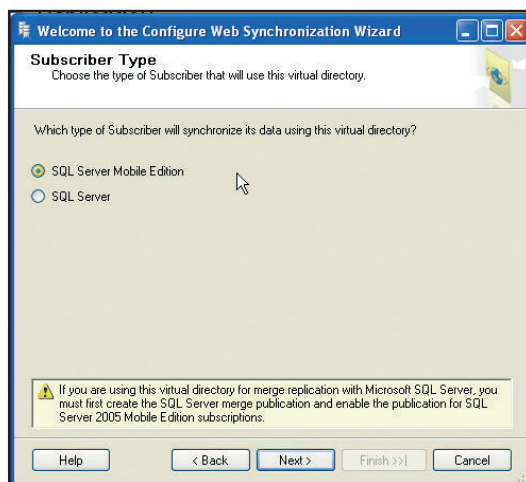
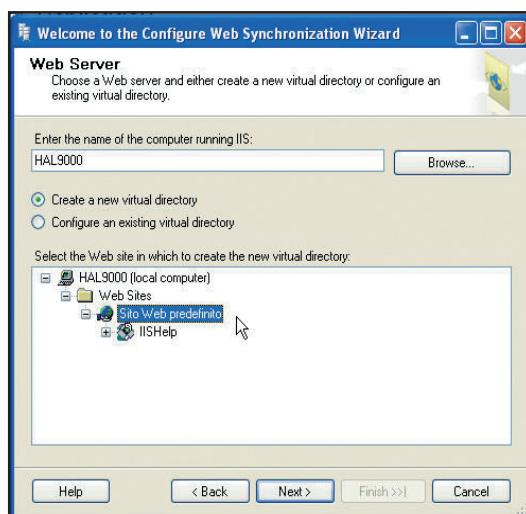
7 Nella pagina Filter Table Rows è possibile aggiungere dei filtri per i dati pubblicati, in modo da diminuire la mole dei dati da trasmettere, nel nostro caso possiamo omettere i filtri ed andare avanti, in modo da visualizzare la pagina Snapshot Agent. Nella pagina Snapshot Agent è possibile creare immediatamente lo snapshot e schedulare la frequenza con cui verrà eseguito l'agente snapshot. Lasciamo le impostazioni predefinite ed andiamo avanti.

8 Nella finestra di dialogo Agent Security possiamo cliccare sul pulsante Security Setting, in questo modo viene mostrata la finestra di dialogo Snapshot Agent Security. In Snapshot Agent Security dobbiamo inserire le informazioni per l'account di dominio che ha accesso alla cartella di Snapshot, pertanto dobbiamo inserire: nome utente di dominio e password di accesso. L'account deve es-

sere scritto nella forma: nome_computer\agente_snapshot. Premendo OK torniamo alla finestra precedente dove possiamo cliccare su Next per andare avanti, lasciamo ancora le impostazioni di default e finalmente arriviamo all'ultima finestra.

9 Nell'ultima schermata Complete the Wizard, possiamo inserire il nome della pubblicazione: TentataVendita e quindi cliccare su Finish. In questo modo viene creata la pubblicazione.

INSTALLARE E CREARE UN DATABASE SQL SERVER MOBILE

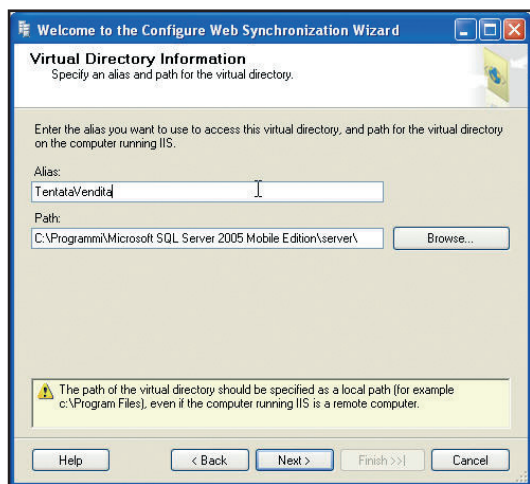
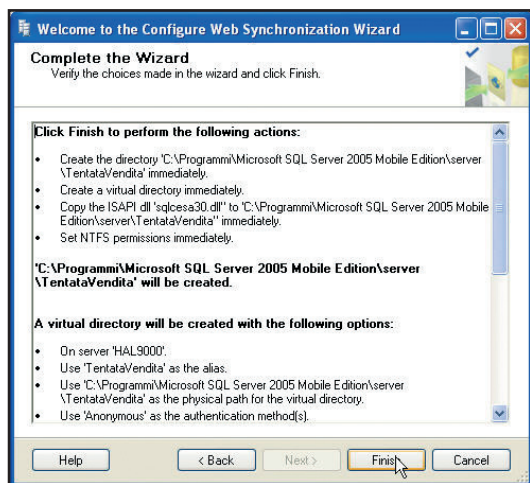
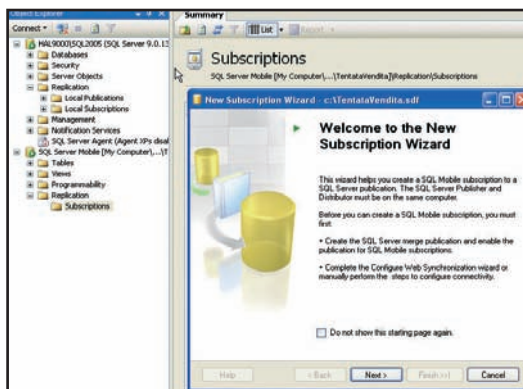
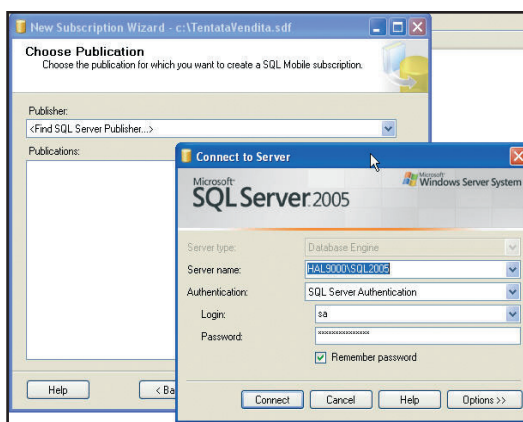
Siamo pronti per installare i componenti server di SqlServer Mobile, che normalmente non vengono installati di default insieme ad Sql Server 2005. La procedura è guidata ed è alquanto banale, basta fare doppio clic sul pacchetto di installazione sqlce30setupen.msi che si trova nella cartella di default C:\Programmi\Microsoft Sql Server\90\Tools\Binn\VSShell\Common7\IDE per eseguire il programma di installazione.

Dopo aver installato SqlServer Mobile, possiamo riavviare Sql Server Management Studio per creare un nuovo database. Nella finestra Object Explorer clicchiamo su Connect e scegliamo l'opzione Sql Server Mobile. Non ci resta che selezionare new database ed assegnarli il nome TentataVendita.sdf, in questo modo verrà mostrato il nuovo nodo che vediamo in figura.

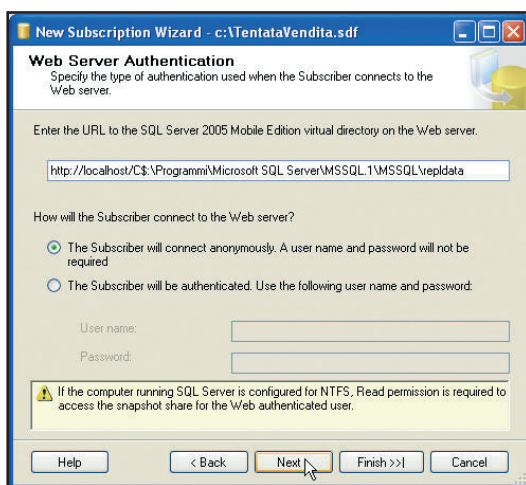
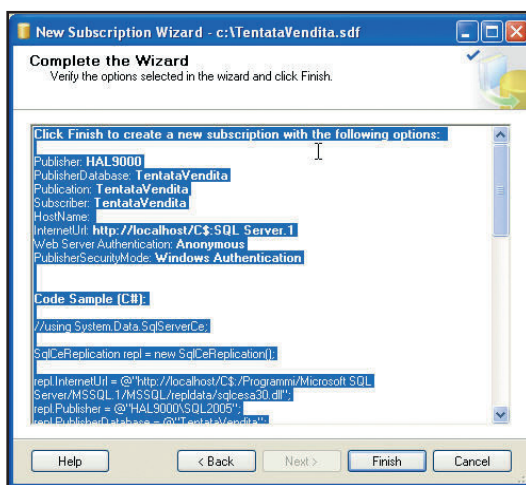
Configurare la sincronizzazione via web e la sottoscrizione

Le ultime due operazioni che ci restano da compiere sono: configurare la pubblicazione per la sincronizzazione via web, creare la sottoscrizione

1 Per configurare la sincronizzazione via web, clicchiamo con il pulsante destro del mouse sulla cartella Replication e selezioniamo la voce Configure Web Synchronization.

Nella solita schermata iniziale della procedura guidata clicchiamo su Next per passare alla schermata successiva. Nella schermata Subscriber Type dobbiamo selezionare l'opzione

SQL Server Mobile Edition ed andare avanti.

2 Nella successiva schermata Web Server, dobbiamo inserire il nome del nostro computer, e selezionare l'opzione Create a new virtual directory
Infine, nella struttura ad albero sottostante, dobbiamo: espandere il computer, espandere la cartella Web Sites e quindi selezionare la voce Sito Web predefinito.

3 Nella schermata Virtual Directory Information dobbiamo immetter l'Alias che andremo ad utilizzare per accedere alla Virtual Directory. Nelle due schermate successive possiamo lasciare le opzioni di default ed andare avanti

4 Nella schermata snapshot share access possiamo immettere il nome della cartella condivisa nella forma: \\computer\snapshot, dove computer rappresenta il nome del computer in uso, ed andare avanti. La prima volta verrà mostrato un avviso che ci dice che la condivisione snapshot è vuota, clicchiamo su Yes per andare avanti. Infine ci viene mostrata l'ultima schermata Complete the Wizard in cui possiamo cliccare su finish per terminare il processo .
Vediamo ora come creare la sottoscrizione

1 Per creare la sottoscrizione dobbiamo visualizzare la finestra Object Explorer, espandere il nodo SQL Server Mobile, espandere il nodo Replication, cliccare con il pulsante destro del mouse sulla cartella Subscription e selezionare la voce New Subscription. Nella solita schermata iniziale possiamo cliccare su Next.

2 Nella schermata Choose Publication dobbiamo selezionare la voce <Find Sql Server Publisher> dalla casella di riepilogo a discesa Publisher, in questo modo viene mostrata la finestra di connessione. Nella finestra di connessione al server, possiamo immettere le informazioni per connetterci al computer locale e quindi cliccare su Connect.

3 Nella schermata Choose Publication possiamo selezionare la nostra pubblicazione dall'elenco delle pubblicazioni disponibili, ed andare avanti. Nella successiva schermata inseriamo il nome della sottoscrizione ed andiamo avanti. Arriviamo così alla schermata Web Server Authentication. In Web Server Authentication dobbiamo inserire l'URL della directory virtuale creata in prece-

denza.
Selezioniamo l'opzione di Connessione anonima ed andiamo Avanti.

4 Lasciamo le impostazioni di default nella successiva schermata relativa all'autenticazione di SQL Server ed arriviamo alla schermata finale.
Nell'ultima schermata Complete the Wizard viene visualizzato il codice di esempio che ci sarà utile nel proseguo dell'articolo quando dovremo creare la sottoscrizione nell'applicazione VB.

ATTENTI A QUEL VIDEO!

Nello sviluppo di applicazioni per dispositivi mobili, si deve porre particolare attenzione al disegno dell'interfaccia. Il fattore determinante è la piccola dimensione del display (tipicamente 320*250) da cui non possiamo considerare lo spazio necessario per visualizzare la barra superiore e quella dei menu in basso. In questo spazio limitato, dovranno trovare posto tutte le informazioni più utili per il corretto funzionamento dell'applicazione.
L'altro fattore da tenere in considerazione, è che l'utente non sarà seduto davanti ad un PC con tastiera e mouse a disposizione ma avrà davanti uno schermo sensibile al tocco e ad un puntamento mediante pennino.

REMOTE DATA ACCESS

In una applicazione VB, per la sincronizzazione dei dati si può utilizzare RDA che permette di accedere in modo semplice ai dati archiviati in un database remoto di Microsoft SQL Server 2000 o SQL Server 2005. Il meccanismo è abbastanza semplice:

- Come primo passo, si effettua la propagazione dei dati nel client.
- Successivamente, viene eseguito il pull dei dati di una tabella dal server al client
- Quindi viene eseguito il push delle modifiche apportate nel client, dal client al server.
- Infine, per aggiornare il client con le nuove modifiche dal server, è necessario eliminare la tabella ed eseguirne nuovamente il pull dal server.

Se una qualsiasi modifica fallisce, gli errori vengono scritti nella error table (che deve essere specificata durante il Pull). Specificando l'opzione BatchingOn, viene fatto un roll back di tutto, altrimenti con BatchingOff, viene fatto un roll back solo della modifica che ha generato l'errore

Riassumendo:

- *Pull* permette di estrarre i dati da un database di SQL Server e di archiviarli in un database di SQL Server 2005 Mobile Edition.
- *Push* permette di inviare le modifiche da una tabella di database di SQL Server Mobile a un database esistente di SQL Server.

I dati possono arrivare da una tabella, una vista o da una stored procedure che restituisce un set di righe, i risultati vengono trasmessi al dispositivo dove sono archiviati in una tabella. Alla richiesta dell'applicazione, le righe aggiornate vengono nuovamente inviate al server, dove vengono applicate al database di SQL Server.
È possibile, ed anche auspicabile, filtrare i dati per ridurre la dimensione del set di righe da aggiornare.
La funzionalità RDA non deve essere configurata sul server ma è necessario configurare singolarmente i client. Infine RDA non supporta i sistemi di risoluzione dei conflitti (ad esempio la modifica dei dati da parte di due utenti diversi) pertanto è necessario scrivere del codice nell'applicazione sul dispositivo per ge-

stire gli errori, è anche possibile registrare gli errori in un'apposita tabella nel database di SQL Server Mobile.

RITORNO AD HELLO WORLD

Torniamo alla nostra applicazione e rendiamola più complessa aggiungendo le funzionalità di accesso al database TentataVendita. Come prima operazione, dobbiamo aggiungere al progetto, un riferimento alla libreria *System.Data.SqlServerCe.*

Nella finestra del codice scriviamo, come prima istruzione:

```
Imports System.Data.SqlServerCe
```

Siamo pronti per aggiungere una connessione dati al database creato in precedenza, selezionando la voce: *Aggiungi nuova origine dei dati* dal *menu Dati* e scegliendo il *database Microsoft SQL Server Mobile Edition*.
Dopo aver effettuato la connessione al database, possiamo mostrare i dati della tabella tblClienti in form1, seguendo questi semplici passi:

1 Dal menu Dati, selezioniamo la voce Mostra origini dati.

2 Nella finestra Origini dati saranno elencate tutte le tabelle contenute nel database, selezioniamo la tabella tblClienti e trascianiamola in Form1. In questo modo verrà creata una griglia dei dati con i nomi delle colonne.

3 Clicchiamo sulla piccola freccia nell'angolo superiore destro della griglia dei dati, e dall'elenco a tendina, selezioniamo l'opzione Genera form dati, in questo modo verranno generate automaticamente le form in cui sarà possibile inserire dei nuovi clienti

Passiamo ora a scrivere il codice necessario alla sincronizzazione dei dati.
Definiamo una variabile stringa che dovrà contenere il nome ed il percorso del file del database:

```
Dim NomeFileDB As String
NomeFileDB =
("\Programmi\TentataVendita\TentataVendita.sdf")
```

A questo punto dobbiamo scrivere una procedura che elimini il file del database, se quest'ultimo esiste già, in questo modo l'applicazione caricherà i dati più recenti ogni volta che viene eseguita.

```
Private Sub CancellaDB()
If System.IO.File.Exists(NomeFileDB) Then
 System.IO.File.Delete(NomeFileDB)

End If
End Sub
```

Infine, dobbiamo scrivere un nuovo metodo (possiamo chiamarlo Sincronizza) per eseguire la sincronizzazione, allo scopo, possiamo utilizzare il codice che abbiamo copiato precedentemente dalla Creazione guidata. Ricordiamo soltanto di apportare le due seguenti modifiche:

- Cambiamo il valore di *SubscriberConnectionString* in modo che punti al percorso e al nome di file corretti, come indicato in *NomeFileDB*.
- Cambiamo il valore di *AddOption* da *ExistingDatabase* a *CreateDatabase*.

A questo punto non ci resta che richiamare i due metodi appena creati, avviare il progetto ed il gioco è fatto.

Luigi Buono

SVILUPPARE UNA CHAT CON AJAX E PHP

IN QUESTA PUNTATA APPROFONDIREMO L'ANALISI DELLA NOSTRA CHAT IN PHP E AJAX INTRODUCENDO NUOVE PROBLEMATICHE, QUALI LA VALIDAZIONE DEI CAMPI DEL FORM DI ISCRIZIONE E LA GESTIONE DEGLI UTENTI

REQUISITI

Conoscenze richieste
PHP 4, MySql, elementi di Javascript e Ajax

Software
PHP 4 o sup., MySql, Apache

Impegno

Tempo di realizzazione

Nella prima puntata di questa serie ci siamo soffermati, in modo particolare sugli strumenti messi a disposizione da Ajax per consentire lo sviluppo di una chat funzionante senza il “reload” del riquadro dei messaggi. È ora venuto il momento di fare qualche passo avanti per rendere la nostra chat decisamente più completa.

Prima di addentrarci nei meandri della programmazione Ajax, è però il caso di soffermarci ancora su alcuni aspetti interessanti relativi al funzionamento della maggior parte delle chat presenti sul Web. La maggior parte di queste chat si basa molto pesantemente su tecnologie Java e Macromedia Flash. Noi invece utilizzeremo solio HTML ed Ajax, per cui è importante capire quali sono le funzionalità che vogliamo aggiungere al nostro software. Immaginiamo di voler realizzare una chat in Java, a tale scopo potremmo utilizzare un’applet che si collega ad un “servizio” attivo sul server, il quale smista i messaggi e li restituisce ai vari client connessi in quel momento (con o senza un’elaborazione preventiva). Il server, per consentire l’accesso di un nuovo utente, resta in ascolto all’infinito su una determinata porta, e quando riceve una chiamata si limita a registrarla (ad esempio creando una nuova istanza della classe Utente che viene immagazzinata assieme alle altre in un oggetto Vector, che per Java rappresenta un array dalle dimensioni non preimpostate).

La creazione del legame di comunicazione Tcp-Ip con i client è assicurata mediante l’utilizzo delle classi Socket e ServerSocket e a quelle relative allo streaming, che si possono avere importando i package **java.io, java.net**:

```
class servizio {
        static int porta=1000;
        public static Vector utenti = new Vector();
        /* l'oggetto Vector utenti è definito con il
        /* modificatore static, questo lo rende
        /* indipendente dalla creazione
        // di una particolare istanza: in effetti,
        // esso dovrà fungere da "contenitore"
        // di istanze

        public static void main(String args[]){
                try{
                        ServerSocket s = new
                          ServerSocket(porta);
        // siccome la condizione
        // true si verifica sempre, il ciclo sotto
        // riportato continua all’infinito,
        // restando in ascolto della porta 1000
        while(true){
            System.out.println("Entrato utente\n");
        // questa scritta verrà visualizzata
        // sulla console del server
          Socket  nuovo_utente = s.accept();
        Utente utente  = new
      Utente(nuovo_utente, "Client" + utenti.size);
                        }
        // quando avvio il  servizio,
        // semplicemente, viene generato
        // un nuovo oggetto
        // Vector che dovrà contenere le varie
        // istanze di utenti collegate alla chat,
        // e quindi il servizio si
        // mette in ascolto della porta 1000
                }catch(Exception e){
                        System.out.println(e);
                }
        }
}
```

La classe Utente, che qui ci siamo limitati ad accennare per ragioni di semplicità, tra le cose “istanzierà” un thread incaricato di restare in ascolto di eventuali nuovi messaggi, attraverso un ciclo infinito simile a quello appena illustrato. I thread consentono al server di ricevere la richiesta da parte di più client contemporaneamente, aspetto essenziale in una chat che deve sempre servire una molteplicità di utenti. Il multithreading si realizza attraverso l’esecuzio-

ne di certe funzioni in parallelo con lo svolgimento di altre attività, similmente ai processi che girano nei sistemi operativi, dando l'illusione della "contemporaneità" del verificarsi di tali azioni (in realtà, è invece il sistema stesso che riesce ad inoltrare alla CPU le varie richieste in modo intelligente, alternandole in modo da non privilegiarne una rispetto alle altre).
L'applet del client, dal canto suo, all'atto del caricamento, aprirà un canale Socket con il server e si preoccuperà di istanziare un thread "sentinella", che resta in ascolto anch'esso di eventuali nuove comunicazioni.
È evidente che tutto ciò genera innumerevoli problemi di basso livello, legati a molteplici aspetti: innanzitutto il programmatore deve lavorare direttamente con il canale Socket, inoltre deve preoccuparsi in prima persona di gestire i problemi relativi alla sicurezza delle applicazioni mutithreading.
Ma le difficoltà che si incontrano tentando l'approccio con Java non riguardano soltanto la fase di progettazione. Anche l'utente finale, infatti, si rende presto conto che tale soluzione "pesa" molto in termini di risorse, comportando una certa lentezza e talvolta persino instabilità nell'utilizzo dell'applicazione. Non dimentichiamo, inoltre, che per eseguire l'applet, essendo essa un vero e proprio programma che "gira" sul client, il browser necessita della Java Virtual Machine, l'unico interprete specializzato del linguaggio Java. Questo significa che per poter utilizzare la nostra piccola chat dobbiamo per forza installare sul pc in locale un linguaggio nuovo, con il suo motore, le sue librerie, ecc…
Ajax, al contrario, non è un linguaggio, né una nuova tecnologia, ma sfrutta semplicemente una nuova concezione della programmazione web-based che utilizza un componente già presente, peraltro, nella maggior parte dei browser in modo nativo: XmlHttpRequest.
Tutto ciò che occorre, per far girare un'applicazione che fa uso di tale componente (o l'ActiveXObject corrispondente nel caso di Internet Explorer), è semplicemente il supporto a Javascript abilitato.
In realtà anche con Ajax, comunque, non sono tutte rose e fiori, soprattutto quando l'applicazione che si intende realizzare richiede un'interazione complessa con il server. L'approccio adottato dalle applet Java, in questi casi, è sicuramente più performante, visto che fa riferimento a veri e propri canali Socket e non a semplici scambi di comunicazioni tra client e server: ma la soluzione Java è anche più efficiente per via della presenza sul server di una memoria persistente condivisa tra tutti i client, che quindi possono tranquillamente attingervi senza dover per forza scomodare funzionalità esterne di archiviazione. PHP invece, a tutt'oggi, non offre la possibilità di condividere delle risorse (come, ad esempio, una variabile) tra più client, quindi effettuando una connessione ad una pagina PHP, tramite Ajax, al più si può riuscire ad aggiornare la propria pagina web, ma per rendere visibili tali modifiche anche a tutti gli altri utenti occorre salvarle, ad esempio, in un file di testo, o in un DB.
Questo genera sicuramente problemi di programmazione non indifferenti: se è vero, infatti, che un file di testo batte sicuramente un DB relazionale in termini di rapidità di inserimento di nuovi informazioni, è vero anche che una simile scelta comporta la necessità di **sincronizzare** l'accesso a tale file (soprattutto in fase di scrittura) da parte dei vari utenti collegati, per evitare crash del sistema o più semplicemente risultati inattesi.
Ma è importante altresì sottolineare che i moderni database relazionali offrono ormai caratteristiche di efficienza e sicurezza straordinarie, oltre a facilitare enormemente il reperimento delle informazioni in essi contenute. La facilità di interrogazione di MySql, in particolare della versione 5 che supporta anche stored procedure e trigger, unitamente alla grande velocità di tale RDBMS, fa pendere sicuramente la bilancia a favore di quest'ultimo, rispetto invece all'utilizzo dei file di testo, che rappresentano comunque una soluzione da tenere nella massima considerazione in molti altri casi.
Abbiamo analizzato, finora, tutti i pregi e i difetti della soluzione Ajax, paragonata ad una equivalente basata su Java: è giunto quindi il momento, finalmente, di cominciare a studiare il funzionamento della nostra chat in PHP, Ajax e MySql.

COME UTILIZZARE I FILE DI ESEMPIO

Sul CD allegato è disponibile la chat di esempio. Per far funzionare correttamente l'applicazione occorre innanzitutto creare il DB e utilizzare il file SQL allegato per popolarlo, quindi modificare le stringhe di connessione con i propri parametri.

COME FARE PER OTTIMIZZARE LE QUERY ?

L'operazione di esecuzione di una query sul DB richiede la creazione, tutte le volte, di un nuovo processo, che causa di solito un certo dispendio di risorse da parte del server. Per ovviare a tale inconveniente, esiste la possibilità di includere la stessa query all'interno di una stored procedure, che semplicemente, dopo essere stata eseguita la prima volta, rimane in memoria in modo permanente evitando, nelle chiamate successive, controlli ulteriori (ad esempio, quelli che verificano la correttezza "grammaticale" della query stessa). In modo analogo, se si sa che una determinata query verrà eseguita ciclicamente nella stessa funzione, si può includere la query in un comando preparato, che compila la struttura della query e ripropone, in occasione delle chiamate successive, la versione compilata, molto più veloce. L'uso di stored procedure e comandi preparati, sicuramente, rappresenta un valido sistema per migliorare le prestazioni di interrogazione del DB senza saturare inutilmente il server.

L'ISCRIZIONE

Prima di poter accedere alla chat dobbiamo necessariamente creare un utente e inserirlo nel database: a tale scopo utilizzeremo la pagina Iscriviti.php, che ci presenta un form da riempire con i nostri dati. È inutile dire che la validazione dei campi inseriti nel form verrà effettuata con Ajax, utilizzando una logica simile a quella adottata per la gestione della chat: su pressione del pulsante Invia i campi vengono inviati al server, senza ricaricare la pagina, il quale effettua tutti i controlli del caso (esistenza dell'utente da noi scelto, immissione di dati non regolari o vuoti), e in caso di esito positivo effettua la registrazione nel DB. In caso contrario, invece, il server si limiterà a riempire con delle segnalazioni di errore i div (inizialmente vuoti) posti accanto ai campi che hanno generato gli errori stessi.
C'è da dire, a questo proposito, che uno dei possibili inconvenienti di Ajax è, paradossalmente, proprio il suo principale punto di forza: ovvero, la grande trasparenza con cui vengono effettuate le richieste al server e catturate le risposte da esso generate. Tale inconveniente si manifesta, in genere, quando la connessione è particolarmente lenta, e c'è il rischio da parte dell'utente di non accorgersi dell'esistenza di un'elaborazione in corso. La possibilità, offerta dal Javascript asincrono, di sollecitare la pagina web anche durante una transazione con il server costituisce, peraltro, un grave rischio che può seriamente compromettere il corretto funzionamento dell'applicazione.
Per rimediare, quindi, a questo comportamento così discreto e taciturno di XmlHttp Request, applicheremo pertanto una sorta di "preload" visualizzando, soltanto in fase di ricezione naturalmente, un'immaginetta animata, che ci rende subito partecipi della "trattativa" con il server in corso. Terminata la fase di ricezione, ovviamente, rimuoveremo nuovamente tale immagine.
Provando la chat in locale, forse, l'immagine di "preload" si vedrà appena, ma collegandosi invece in remoto per utilizzare l'applicazione magari in contemporanea con più utenti, ci si rende immediatamente conto dell'utilità di tale accorgimento.
Ecco la funzione Javascript che invia i campi al server per la validazione:

```
function valida()
{
    if (xmlHttp)
    {
        if (xmlHttp.readyState == 4 ||
                xmlHttp.readyState == 0)
        {
nick=encodeURIComponent(document.
            getElementById("nick").value);
nome=encodeURIComponent(
        document.getElementById("nome").value);
cognome=encodeURIComponent(document.
        getElementById("cognome").value);
email=encodeURIComponent(document.
        getElementById("email").value);
telefono=encodeURIComponent(document.
        getElementById("telefono").value);
citta=encodeURIComponent(document.
        getElementById("citta").value);
indirizzo=encodeURIComponent(document.
        getElementById("indirizzo").value);
provincia=encodeURIComponent(document.
        getElementById("provincia").value);
            try
            {
document.getElementById("loading")
            .style.visibility="visible";
            xmlHttp.open("POST",
                    "valida.php", true);
xmlHttp.setRequestHeader("Content-Type",
            "application/x-www-form-urlencoded");
            xmlHttp.onreadystatechange =
                    risposta_validazione;
            xmlHttp.send("nick=" + nick +
  "&nome=" + nome + "&cognome=" + cognome +
    "&email=" + email + "&telefono=" + telefono +
     "&citta=" + citta + "&indirizzo=" + indirizzo +
                "&provincia=" + provincia);
            }
            catch (e)
            {
                alert ("Errore nella validazione dei
        campi... Riprova tra qualche secondo ...");
                setTimeout("valida();", 5000);
            }
        }
    }
}
```

Come si può vedere, in tale funzione viene reso visibile il *div* contenente l'immagine di caricamento. Lo stesso *div* viene poi reso invisibile al termine della funzione asincrona *risposta_validazione()*, attraverso l'istruzione:

```
document.getElementById("loading").
        style.visibility = "hidden";
```

Un cenno a parte merita, infine, il metodo per recuperare la risposta generata dal server: in questo caso non mi sono avvalso del parametro *responseXml*, come per il resto della chat, ma di *responseText*. Quindi la risposta sarà trattata come una semplice stringa testuale, che viene

splittata del gruppo di caratteri [|] usato per separare le varie coppie **nome_del_div => valore**. Con un semplice ciclo, quindi, aggiorniamo il contenuto dei vari div corrispondenti ad ogni segnalazione di errore.
Ecco la parte di codice che esegue il parsing della stringa:

```
if(response.indexOf('[|]' != -1)) {
  update = response.split('[|]');
    while (update[counter])
   {
document.getElementById(update[counter]).inn
erHTML = update[counter+1];
    ...
```

L'utente registrato può, a questo punto, entrare direttamente nella chat, loggandosi con il proprio nickname scelto: l'invio di un utente non presente nel DB o di un utente già loggato solleverà degli errori gestiti, in questo caso, per semplici ragioni di opportunità di programmazione, con PHP (anche se la stessa funzionalità potrebbe, tranquillamente, essere effettuata tramite Ajax). Se la fase di login va invece a buon fine, l'utente verrà immediatamente reindirizzato alla pagina che ospita la chat.

UTILIZZO DELLA CHAT

Finalmente, è possibile ora chattare liberamente con i vari utenti collegati. La logica della chat in Ajax si basa, in questo caso, sull'esistenza di un array globale di Javascript nel quale vengono immagazzinati i vari messaggi da inviare al server. Il funzionamento di tale array è presto sperimentato: provate a modificare il tempo di timeout (ossia l'intervallo di tempo esistente tra una richiesta al server e l'altra), impostando la variabile *intervallo* del foglio Chat.js, per esempio, a 5000 (5 secondi). Provate ora a inviare molti messaggi in rapida successione, entro i 5 secondi prestabiliti: vedrete che i tanti messaggi che abbiamo cercato di inviare non si sono persi, ma ad uno ad uno vengono reinviati al server allo scadere di ogni intervallo. Questo sistema ha il duplice vantaggio di rispettare il giusto ordine delle richieste effettuate (queste infatti vengono estratte a partire dalla più vecchia, secondo il metodo del FIFO -> First In, First Out), e di assicurare che neppure una di esse vada persa.
Ogni richiesta al server ha lo scopo di catturare i nuovi messaggi non ancora presenti nella nostra lista, e di aggiornare la situazione degli utenti. Mentre i messaggi vengono semplicemente aggiunti alla lista (quindi sarà più che sufficiente intercettare quelli che hanno id superiore all'id dell'ultimo messaggio scaricato, memorizzato in una variabile), la situazione degli utenti potrà subire variazioni notevoli nel corso della chat (è possibile che entrino nuovi utenti, ma è anche possibile che qualcuno esca). In ogni caso non è necessario ricostruire tutte le volte il riquadro degli utenti: basta invece richiedere la situazione aggiornata di volta in volta al server e poi confrontarla con quella che abbiamo nella nostra pagina (salvata, ad esempio, in una variabile, o ricavata direttamente interrogando la proprietà innerHTML del controllo): se c'è qualche differenza, allora dobbiamo ridisegnare la lista, altrimenti semplicemente non facciamo nulla.
E come facciamo per gestire l'uscita di un utente? Potremmo tranquillamente utilizzare l'evento *onunload*, per mandare una richiesta al server non appena l'utente chiude la pagina. Tuttavìa, tale evento non si scatena soltanto con la chiusura della pagina, ma anche facendo il semplice "refresh" della stessa. Siccome, inoltre, non disponiamo di un canale di comunicazione continuo con il server (HTTP, come citavamo nel precedente articolo, è un protocollo privo di stato), dobbiamo quindi creare un meccanismo che si occupi da solo di cancellare gli utenti che non mandano più richieste al server per un certo periodo. Ipotizzando, ad esempio, che ogni 2 secondi venga effettuata una connessione per scaricare i nuovi messaggi e aggiornare la situazione degli utenti, possiamo anche approfittare dell'occasione per aumentare la nostra "vita" all'interno della chat di altri 5 secondi: in pratica è come se comunicassimo agli altri membri della chat che per almeno altri 5 secondi noi siamo sicuramente collegati. Se un utente non manda questa comunicazione passati i 5 secondi (ma in generale dovrebbe rinnovarla già nei prossimi 2), perché ad esempio ha chiuso la sua chat, allora sicuramente egli verrà rimosso dalla tabella degli utenti in

DOVE POSSO APPROFONDIRE GLI ARGOMENTI TRATTATI ?

La tecnica adottata da questo tutorial, che utilizza XmlHttpRequest in modo diretto, rappresenta una scelta da valutarsi attentamente a seconda della grandezza e complessità del progetto che si intende realizzare. Tra i libri che seguono lo stesso approccio, mi sento sicuramente di consigliare l'ottimo "Ajax and PHP" (Editore Pack Publishing), ricco di esempi pratici molto validi e concreti. Per progetti di dimensioni maggiori, tuttavìa, consiglio piuttosto l'utilizzo di framework specializzati: in particolare, in questo momento, si sta facendo avanti JSON, una libreria che consente un approccio notevolmente facilitato ad Ajax e prestazioni decisamente superiori rispetto, per esempio, ad XML.

chat. Esclusivamente in occasione dell'ingresso nella chat, inoltre, darei all'utente subito una "vita" di 10 secondi, visto che non si è ancora completato lo scaricamento della pagina e c'è, quindi, il rischio che egli possa essere involontariamente cancellato prima che parta l'evento **onload** (quello, per intenderci, che si scatena non appena il documento è stato scaricato completamente e che, nel nostro caso, effettua la connessione al server tramite Ajax).
Ecco, a titolo di esempio, la porzione di codice che gestisce la scadenza degli utenti:

```
$query = "update utenti_in_chat set ora = now()
        + interval 5 second where nickname = '" .
            mysql_escape_string($utente) . "'";
$risultato=mysql_query($query) or die($query);
//cancello gli utenti che
//non mandano più richieste
                    al server da un certo tempo
    $query = "select * from utenti_in_chat
                        where ora < now()";
    $risultato=mysql_query($query) or
                                 die($query);
    if (mysql_num_rows($risultato)>0)
    {
        while ($riga1 =
                  mysql_fetch_array($risultato))
        {
                $nick_utente =
    mysql_escape_string($riga1['nickname']);
                $query="delete from
          utenti_in_chat where nickname='"
                       . $nick_utente . "'";
$risultato1=mysql_query($query) or die($query);
                $messaggio='Sono
                  appena uscito dalla chat';
                $query = 'insert into
           messaggi(utente, messaggio, ora) ' .
                              'values ("' .
        $nick_utente . '" , "' . $messaggio .
                               '", now())';
    $risultato2=mysql_query($query) or
                                 die($query);
        }
    }
```

Questo meccanismo andrebbe sicuramente affinato meglio, per gestire, ad esempio, la scadenza della sessione di utilizzo della chat da parte degli utenti inattivi da un certo periodo. Anche i controlli sull'ingresso di un nuovo utente in chat andrebbero migliorati, in modo da evitare la visualizzazione del messaggio di disconnessione dell'ultimo utente presente nella tabella utenti_in_chat quando la differenza tra l'ora attuale e quella di registrazione supera un certo periodo prestabilito. Il controllo dell'esistenza di un utente nella tabella utenti_in_chat, inoltre, dovrebbe impedire nuovi accessi da parte dello stesso utente soltanto se la differenza tra l'ora attuale e quella di registrazione supera a un certo limite fissato a priori. Di sicuro, quindi, la logica accennata finora è molto migliorabile, ma rappresenta già così un valido sistema per gestire in modo semplice ed efficace la "vita" degli utenti all'interno della chat.

CONCLUSIONI

Per concludere questa trattazione, mi sembra doveroso un accenno alle precauzioni da adottare per evitare di saturare il server con troppe chiamate inutili. Innanzitutto, è bene fissare degli intervalli di chiamata del server abbastanza distanti tra di loro; effettuare "refresh" della pagina ogni mezzo secondo è decisamente inutile, oltreché dannoso: ricordiamoci, infatti, che tutte le volte che chiamiamo il server eseguiamo anche delle query, che appesantiscono molto il sistema. Sicuramente l'utilizzo di comandi sql preparati o stored procedure (che con MySql 5 sono finalmente disponibili), unitamente a qualche sistema personalizzato di caching delle informazioni sul client, migliora notevolmente le prestazioni, riducendo i colli di bottiglia ed evitando inutili sprechi di risorse sul server.
Naturalmente Ajax non è la soluzione ad ogni problema e certamente la soluzione da noi presentata non può avere le stesse performance o dotazioni di sicurezza tipiche di applicazioni client/Server bastae su java o .net. Tuttavia proprio grazie a questa soluzione possiamo evitare di installare un qualunque servizio sulla macchina che hosta il nostro sito operazione quasi sempre non permessa dai gestori. Inoltre la facilità di implementazione è sicuramente maggiore rispetto a quella che potremmo ottenere dovendo riprogrammare una qualunque architettura client server. Naturalmente la scelta dipende esclusivamente dalle vostre esigenze e disponibilità. Tuttavia oggi è possibile risolvere in questo modo una vasta casistica di problemi.

Enrico Viale

L'AUTORE

Enrico Viale è specializzato nello sviluppo di applicazioni sia web-oriented che desktop in ambiente Windows. Chi desidera contattarlo per chiarimenti riguardo all'articolo, o per qualsiasi altro motivo, può farlo all'indirizzo enrico.viale@gmail.com.

A COSA SERVE LA FUNZIONE HTMLSPECIALCHARS ?

La funzione htmlspecialchars è utile per evitare che una stringa contenente tag html venga interpretata dal browser come tale. I tag, infatti, non vengono neppure eliminati dalla stringa, ma vengono trattati come se non avessero il significato particolare che hanno di solito per il browser.

I trucchi del mestiere

Tips & Tricks

Questa rubrica raccoglie trucchi e piccoli pezzi di codice, frutto dell'esperienza di chi programma, che solitamente non trovano posto nei manuali. Alcuni di essi sono proposti dalla redazione, altri provengono da una ricerca su Internet, altri ancora ci giungono dai lettori. Chi volesse contribuire, potrà inviare i suoi Tips&Tricks preferiti. Una volta selezionati, saranno pubblicati nella rubrica. Il codice completo dei tips è presente nel CD allegato nella directory *tips*\ o sul Web all'indirizzo: cdrom.ioprogrammo.it.

VISUAL BASIC

ORA DI SISTEMA

Come posso settare l'ora di sistema?
Imposta una struttura SYSTEMTIME e passala alla funzione SetSystemTime

```
Dim ora As New SYSTEMTIME
' Ora corrente
GetSystemTime(ora)
ora.wSecond = UInt16.Parse("00")
ora.wMinute = UInt16.Parse("00")
ora.wHour = UInt16.Parse("00")
' modifica gli altri campi se necessario
' Imposta l'ora modificata
SetSystemTime(ora)
```

Per quanto riguarda la SetSystemTime fai riferimento alla libreria Kernel32.dll.

```
<DllImport("kernel32.dll")> _
Shared Function SetSystemTime(ByRef time As SYSTEMTIME)
                                              As Boolean
End Function
```

Si tratta comunque di una soluzione unmanaged
Dal forum di ioProgrammo: http://forum.ioprogrammo.it/thread.php?threadid=10479&boardid=27

PYTHON

RICERCARE PAROLE IN UN FILE

Ciao a tutti il mio problema è il seguente: vorrei far si che data una parola chiave da ricercare in un determinato file di testo ogni volta che essa compare nel file venga stampata a video, spero che qualcuno di voi posso aiutarmi perchè nn sò più dove mettere le mani e purtroppo google nn mi è stato di grade aiuto

Prima di tutto devi dare uno sguardo al modulo fileinput e la gestione delle stringhe in Python. Una soluzione applicativa invece è la seguente:

```
files = glob.glob("*.txt") # Filtro
for riga in fileinput.input(files):
  trovata = string.find(riga, parola)
  if trovata > 0:
        # Stampa
```

dal forum di ioProgrammo http://forum.ioprogrammo.it/thread.php?threadid=10472&boardid=37

.NET

CURSORI ANIMATI

Come posso utilizzare dei cursori animati in C# o VB.NET?

La classe cursor di .NET non supporta direttamente i cursori animati. La documentazione ufficiale MSDN riporta il seguente workaround
C#

```
using System;
using System.Runtime.InteropServices;
using System.Windows.Forms;

namespace Microsoft.Samples.Msdn.Windows {
  public class CursorFactory    {
    [DllImport("user32.dll",
        EntryPoint = "LoadCursorFromFileW",
        CharSet = CharSet.Unicode)]
   private static extern IntPtr LoadCursorFromFile(String str);

   public static Cursor Create(string filename){
      IntPtr hCursor;
      Cursor result = null;
        try {
          hCursor = LoadCursorFromFile(filename);
          if (!IntPtr.Zero.Equals(hCursor)) {
```

```
                result = new Cursor(hCursor);
            } else {
                throw new
                ApplicationException("Could not create cursor from file "
                    + filename);
            }
        } catch(Exception ex) {
            //log exception
        }

        return result;
    }
  }
}
```

Visual Basic .NET

```
Namespace Microsoft.Samples.Msdn.Windows
    Public Class CursorFactory
        Private Declare Unicode Function LoadCursorFromFile _
            Lib "user32.dll" _
            Alias "LoadCursorFromFileW" _
            (ByVal filename As String) As IntPtr

        Public Shared Function _
            Create(ByVal filename As String) As Cursor
            Dim hCursor As IntPtr
            Dim result As Cursor = Nothing

            Try
                hCursor = LoadCursorFromFile(filename)
                If Not IntPtr.Zero.Equals(hCursor) Then
                    result = New Cursor(hCursor)
                Else
                    'could not create cursor
                    Throw New _
                        ApplicationException( _
                        "Could not create cursor from file " & _
                        filename)
                End If
            Catch ex As Exception
                'log exception
            End Try

            Return result
        End Function

    End Class
End Namespace
```

MANDARE EMAIL

Come posso mandare una email in ASP.NET?
Il modo più semplice è il seguente;

```
            try
        {

            SmtpClient smtpMailObj = new SmtpClient();

                //eg:localhost, 192.168.0.x, replace qui ci va il nome
del server

            smtpMailObj.Host = "myMailServer";

            smtpMailObj.Send(txtFrom.Text, txtTo.Text, txtSubject.Text,
txtComment.Text);

            Response.Write("Your Message has been sent successful-
ly");

        }

        catch (Exception ex)

        {

            Response.Write("Message Delivery Fails");

        }
```

Usando l'oggetto MailMessage

```
        MailMessage Mail = new MailMessage();
        MailAddress ma = new
MailAddress(txtFrom.Text,txtName.Text);
        Mail.From = ma;
        Mail.To.Add(txtTo.Text);
        Mail.Subject = txtSubject.Text;
        Mail.Body = txtComment.Text;
        try
        {

            SmtpClient smtpMailObj = new SmtpClient();
            smtpMailObj.Host = "myMailServer";
            smtpMailObj.Send(Mail);
            Response.Write("Your Message has been sent
                                                successfully");
        }

        catch (Exception ex)
        {
            Response.Write("Message Delivery Fails");
        }
```

FILE REMOTI

Come posso recuperare il contenuto di un file HTML direttamente da Web?

```
<?php
$file = fopen ("http://www.example.com/", "r");
if (!$file) {
  echo "<p>Impossibile aprire il file remoto.\n";
  exit;
}
while (!feof ($file)) {
  $linea = fgets ($file, 1024);
  /* Funziona solo se title e i relativi tag sono sulla medesima riga
*/
  if (eregi ("<title>(.*)</title>", $linea, $out)) {
     $title = $out[1];
     break;
  }
}
fclose($file);
?>
```

GESTIRE IIS

Posso gestire IIS direttamente da PHP?

Certamente. E' necessario abilitare l'estensione php_iisfunc.dll all'interno di PHP.ini. In questo modo si avranno a disposizioni funzioni del tipo:

iis_add_server -- Crea un nuovo web server virtuale
iis_get_dir_security -- Riceve informazioni sulla sicurezza della cartella
iis_get_script_map -- Riceve informazioni sul mapping dello script sulla cartella virtuale per una specifica estensioneo
iis_get_server_by_comment -- Restituisce il numero d'istanza associato con il commento
iis_get_server_by_path -- Restituisce il numero di istanza associato alla Path
iis_get_server_rights -- Riceve informaizoni sui diritti del server
iis_get_service_state -- Avvia il servizio definito da service_id
iis_remove_server -- Rimuove il web server virtuale indicato da Server_instance
iis_set_app_settings -- Crea application scope per una cartella virtuale
iis_set_dir_security -- Imposta la sicurezza nella cartella
iis_set_script_map -- Imposta il mapping dello script sulla cartella virtuale
iis_set_server_rights -- Imposta i diritti del server
iis_start_server -- Avvia il web server virtuale
iis_start_service -- Avvia il servizio definito da service_id

SERVER FTP

Posso connettermi ad un server FTP usando PHP?

Si, l'esempio d'uso è il seguente:

```
<?php
// stabilire una connessione
$conn_id = ftp_connect($ftp_server);
// login con user name e password
$login_result = ftp_login($conn_id, $ftp_user_name,
$ftp_user_pass);
// controllo della connessione
if ((!$conn_id) || (!$login_result)) {
      echo "La connessione FTP è fallita!";
      echo "Tentativo di connessione a $ftp_server per l'utente
$ftp_user_name";
      die;
   } else {
      echo "Connesso a $ftp_server, utente $ftp_user_name";
   }
// upload del file
$upload = ftp_put($conn_id, $destination_file, $source_file,
FTP_BINARY);
// controllo dello stato di upload
if (!$upload) {
      echo "Il caricamento FTP non è andato a buon fine!";
   } else {
      echo "Caricato il file $source_file su $ftp_server come $desti-
nation_file";
   }
// chiudere il flusso FTP
ftp_quit($conn_id);
?>
```

USARE I TRIGGER

Cosa è un trigger e come si fa a farlo con MySQL

Un trigger è sostanzialmente una funzione che intercetta un evento su una tabella e reagisce in maniera consequenziale. Ad esempio potrebbe intercettare l'evento insert e riportare questo evento un un file di log. L'esempio è il seguente:

```
mysql> CREATE TABLE account (acct_num INT, amount
                                                  DECIMAL(10,2));
Query OK, 0 rows affected (0.03 sec)
mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account
    -> FOR EACH ROW SET @sum = @sum + NEW.amount;
Query OK, 0 rows affected (0.06 sec)
```

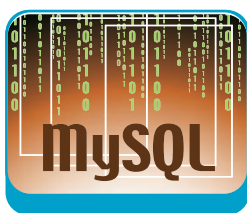
USARE LE REGULAR EXPRESSION

Posso usare le REGEX in una ricerca MySQL?

Si è possibile usare la parola chiave 'REGEXP'. Ad esempio:

```
SELECT * FROM `tabella` WHERE `campo` REGEXP 'regular
                                                  expression'
```

RICERCHE FULL TEXT CON DB MYSQL

MOLTO SPESSO NON SONO SUFFICIENTI LE USUALI RICERCHE DI PARTI DI STRINGA ALL'INTERNO DEI CAMPI TESTUALI: L'ARTICOLO ILLUSTRA COME LE RICERCHE FULL TEXT PERMETTANO INTERROGAZIONI DI TIPO AVANZATO

Chiunque abbia usato l'SQL per l'interrogazione di basi di dati sa che è comune usare l'operatore LIKE per la ricerca di record che contengono determinate parti di stringa. Questo operatore, anche con i caratteri speciali, è limitato per molti aspetti: innanzitutto non permette pattern matching di tipo avanzato e, in secondo luogo, le sue prestazioni non sono ottimali in presenza di numerosi record. Inoltre, cosa non trascurabile, il risultato è dato da record che contengono il termine ricercato ma non c'è la possibilità di ordinare i risultati in maniera da assegnare uno score o un indice di migliore (o peggiore) attinenza. Per esempio, potrebbe essere interessante far sì che i primi risultati siano i record con più match in presenza di più stringhe, e che gli ultimi siano quelli con meno. Questi limiti rendono praticamente inutilizzabile questa soluzione quando la ricerca avviene all'interno di pagine Web (anche all'interno di uno stesso sito, per non parlare di veri e propri motori di ricerca che vorrebbero indicizzare tutto il Web!). È vero che molti DBMS permettono l'uso di espressioni regolari, ma anche questa soluzione non è ottimale (si pensi alla difficoltà ad inserire espressioni regolari per un utente non tecnico, come può essere una form di ricerca sul Web!). Per aggirare questi problemi è stato introdotto il concetto di ricerca "full text".

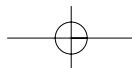
RICERCHE FULL TEXT

Una ricerca di tipo full text si applica su uno o più campi testuali e le query esprimibili possono essere semplici (cercare un determinato termine) o complesse (ricerca di due e più termini, specifica di termini più significativi di altri e, magari, contemporanea assenza di determinati insiemi di termini). Spesso i DBMS che supportano questo tipo di ricerche permettono anche di utilizzare caratteristiche per filtrare le parole ricercate e ottimizzare la velocità di risposta: è comune l'uso di liste di parole da non considerarsi nella ricerca (cosiddette "stop list") come pure per eliminare i caratteri di separazione (virgole, punti e così via); molti offrono anche ricerche per sinonimi, parti di parola o parole simili (numero minimo di caratteri diversi) e così via. Le soluzioni disponibili o sono rese accessibili da caratteristiche speciali direttamente dal DBMS in uso (è il caso che considereremo in MySql, ma esistono alternative simili anche in Oracle e PostgreSQL e molti altri DBMS), oppure è necessario fornire delle soluzioni ad-hoc esterne (per esempio attraverso stored procedure o simili) o, ancora, ricorrere alla creazione di sistemi di ricerca esterni al DBMS stesso (si pensi, per Java, al progetto Lucene, si veda http://lucene.apache.org/). Come si può intuire le ricerche full text, nel loro complesso, rappresentano un argomento di ricerca molto vasto e difficilmente circoscrivibile. Per questo lasciamo da parte la teoria e focalizziamo l'attenzione su esempi reali usando un prodotto concreto: MySql.

TABELLE PER IL FULL TEXT IN MYSQL

Il supporto per le ricerche full text in MySql è limitato alle tabelle di tipo MyISAM e si ottiene usando la parola chiave FULLTEXT nella creazione di una tabella; ecco la tabella di esempio usata nell'articolo

```
CREATE TABLE pensieri_parole (
 id INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
 titolo VARCHAR(50),
 contenuto TEXT,
 FULLTEXT (titolo, contenuto)
)ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

La versione di riferimento è la 5.0.27 (l'ultima stabile al momento di andare in stampa) ma saranno illustrate anche nuove caratteristiche presenti nella versione di sviluppo 5.1.14. Una volta creati gli indici (in questo caso uno solo e comprende due campi; è possibile crearne di nuovi, anche con un campo alla volta o più di due campi), non resta che analizzare le possibili interrogazioni e i modi di sfruttare le ricerche full text. Per i successivi esempi si farà uso di questo insieme di record:

```
INSERT INTO pensieri_parole (titolo, contenuto)
VALUES
 ('preferito','Non c''è un pensiero preferito'),
 ('Confronti','javascript? Più bello del VBScript!'),
 ('Troppi pensieri','Non programmo più'),
 ('Basta parole','Basta parlare di Java'),
 ('Quale linguaggio','Java? bello'),
 ('Linguaggio e parole', 'Se non lo penso non lo so')
;
```

MODALITÀ DI RICERCA BASE

La prima, e più semplice, modalità di ricerca full text consiste nell'usare il costrutto:

```
MATCH(indice) AGAINST('lista di parole')
```

In questa modalità è possibile specificare una lista qualsiasi di parole. Il costrutto può essere usato nella clausola WHERE: in questo caso restituisce tutti i record che hanno almeno una parola all'interno della lista specificata, ordinati per rilevanza. Per esempio:

```
SELECT * FROM pensieri_parole
 WHERE MATCH (titolo, contenuto)
 AGAINST ('linguaggio bello');
```

Il risultato è costituito dalla prima e dall'ultima riga inserite (**Figura 1**).

```
c:\wamp5_1.6.6\mysql\bin\mysql.exe
mysql> SELECT * FROM pensieri_parole
    ->  WHERE MATCH (titolo, contenuto)
    ->  AGAINST ('linguaggio bello');
+----+---------------------+-------------------------------------+
| id | titolo              | contenuto                           |
+----+---------------------+-------------------------------------+
|  5 | Quale linguaggio    | Java? bello                         |
|  6 | Linguaggio e parole | Se non lo penso non lo so           |
|  2 | Confronti           | javascript? Pi¨ bello del VBScript! |
+----+---------------------+-------------------------------------+
3 rows in set (0.03 sec)

mysql>
```

***Fig. 1:** i risultati della prima query fulltext.*

Questo fa capire che non è importante dove si trovino le parole nei diversi campi: è come se l'indice fosse costruito su un campo fittizio, contenente tutti i campi specificati. Lo stesso costrutto può essere inserito nei valori della SELECT: in questo caso vengono reperiti gli score assegnati a ciascuna riga:

```
SELECT id, MATCH (titolo, contenuto)
 AGAINST ('linguaggio bello')
 FROM pensieri_parole;
```

Si può constatare che l'ordine dei risultati della prima query è dato in base allo score (dal più grande al più piccolo) e sono esclusi i record con score il cui valore è zero. In MySql 5.1 la sintassi di ricerca è leggermente diversa:

```
SELECT * FROM pensieri_parole
 WHERE MATCH (titolo, contenuto)
 AGAINST ('linguaggio bello' IN NATURAL LANGUAGE
                                          MODE);
```

Si noti la presenza delle parole chiave "IN NATURAL LANGUAGE MODE".

RICERCHE IN MODALITÀ "BOOLEAN"

Per esprimere una query in modalità boolean il costrutto diventa:

```
MATCH (titolo, contenuto)
 AGAINST ('lista parole' in BOOLEAN MODE);
```

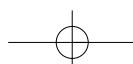
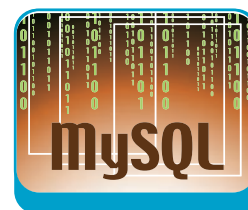
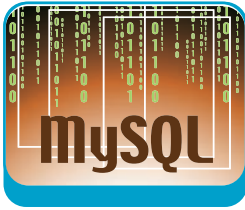
A differenza delle altre ricerche, è possibile specificare un modificatore per ciascuna parola della lista. Tale modificatore va a cambiarne il significato. La lista dei modificatori esprimibili è sintetizzata in **Tabella 1**. Ora si creeranno degli esempi che illustrano l'uso di ciascun modificatore. Per provare queste ricerche, che sono, con tutta probabilità, quelle più evolute, è possibile (come prima) usare la console di comandi di MySql ma anche una delle due applicazioni Web a corredo dell'articolo (una scritta in PHP ed una in JSP, quest'ultima fornita come archivio WAR pronto per il deploy su un Servlet Container come può essere Tomcat). Le applicazioni Web permettono di specificare, su una campo di inserimento dati, la query (in modalità boolean) da eseguire.

Modificatore	Significato
+	Il termine che segue deve essere presente in ogni riga restituita
-	Il termine che segue non deve essere presente in nessuna riga restituita
>	Il termine che segue è considerata rilevante
<	Il termine che segue è considerata poco rilevante
~	Il termine che segue può essere presente ma, in questo caso,
	la sua presenza fa diminuire lo score dell'intera riga
"…"	Cerca (tutti) i termini nell'ordine che sono specificati
(…)	Le parentesi tonde permettono di specificare delle sotto-espressioni
	(è possibile applicare un qualsiasi modificatore anche prima della
	prima parentesi)

***Tab. 1:** i modificatori esprimibili nella modalità "Boolean"*

SENZA MODIFICATORI

Se non c'è alcun modificatore allora, in maniera simile a quanto fatto prima, si cercano record che possono contenere una qualunque delle parole specificate. In realtà i risultati non sono gli stessi; per rendersene conto si può eseguire la stessa query di prima, con la stampa degli score, ma con la nuova modalità:

```
SELECT id, MATCH (titolo, contenuto)
 AGAINST ('linguaggio bello' IN BOOLEAN MODE)
 FROM pensieri_parole;
```

si può osservare che ora gli score sono espressi in valori interi. In pratica, maggiori sono i match maggiore è lo score, ma sempre come numero di occorrenze.

IL MODIFICATORE '+'

Il '+' indica che la parola indicata deve essere presente. Pertanto se si effettua una query di questo tipo:

```
SELECT * FROM pensieri_parole
WHERE MATCH (titolo, contenuto)
AGAINST ('+linguaggio +bello' in BOOLEAN MODE);
```

si otterranno solo i record che contengono sia linguaggio che bello (e non, come prima, anche uno soltanto dei termini della lista).

IL MODIFICATORE '–'

C'è anche la possibilità di specificare che un determinato termine non deve essere presente in alcun record reperito: in questo caso si ricorre al modificatore '-'.
Per esempio:

```
SELECT * FROM pensieri_parole
WHERE MATCH (titolo, contenuto)
AGAINST ('+linguaggio -bello' in BOOLEAN MODE);
```

Come sempre, è possibile conoscere gli score inserendo la clausola MATCH tra i campi reperiti della SELECT e "indagare" sugli score ottenuti:

```
SELECT id, MATCH (titolo, contenuto)
AGAINST ('+linguaggio -bello' IN BOOLEAN MODE)
FROM pensieri_parole;
```

Questo è un ottimo modo per comprendere il motivo di determinati risultati e, in particolare, dell'ordinamento dei record.

IL MODIFICATORE '*'

È bene osservare che la ricerca full text, normalmente, non ricerca parti di stringa, ma solo termini completi. Se, viceversa, si volesse usare anche questa caratteristica, si deve usare il modificatore "*":

```
SELECT * FROM pensieri_parole
WHERE MATCH (titolo, contenuto)
AGAINST ('java*' in BOOLEAN MODE);
```

In questo caso, per esempio, sono reperiti sia record contenenti "Java" che quelli con "JavaScript".

I MODIFICATORI '>' E '<'

Talvolta si vorrebbero includere alcuni termini, ma si sa che sono meno significativi di altri, sempre espressi nella stessa query. In questo caso si può ricorrere ai modificatori di rilevanza: con ">" si indicano i termini più rilevanti, con "<" quelli meno:

```
SELECT * FROM pensieri_parole
WHERE MATCH (titolo, contenuto)
AGAINST ('>java <linguaggio' in BOOLEAN MODE);
```

IL MODIFICATORE '~'

Esiste anche un caso a metà strada tra l'uso del modificatore di esclusione ('-') e quello di minore rilevanza ('<'): infatti il primo esclude i record in toto, il secondo considera il termine, seppur meno rilevante, comunque facente parte dell'incremento dello score in caso di sua presenza. Il caso non trattato è quando si vuol far sì che la presenza del termine diminuisca lo score complessivo, ma includendo comunque quei record con score maggiori di zero. In questi casi si usa l'operatore '~' (tilda). Ecco, per esempio, che la seguente query:

```
SELECT *, MATCH (titolo, contenuto)
AGAINST ('~java linguaggio' in BOOLEAN MODE)
FROM pensieri_parole;
```

fa sì che lo score del quinto record venga abbassato a 0.5!

IL MODIFICATORE "..."

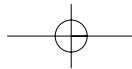
Usando i doppi apici si forna il match di tutti i termini racchiusi. A partire da MySql 5.0.3 vengono ignorati i caratteri separatori. Per esempio la seguente query:

```
SELECT * FROM pensieri_parole
WHERE MATCH (titolo, contenuto)
AGAINST ('"java bello"' in BOOLEAN MODE);
```

Permette di reperire il solo record che contiene "Java? Bello" (mentre se si tolgono i doppi apici vengono reperiti tutti i record che contengono uno qualunque tra "java" e "bello").

IL MODIFICATORE "( )"

Infine è possibile comporre liste diverse di termini e modificatori usando le parentesi tonde; al loro interno è possibile esprimere una qualsiasi sotto-espressione, magari assegnando ad essa un ulteriore modificatore; per esempio:

```
SELECT * FROM pensieri_parole
WHERE MATCH (titolo, contenuto)
```

```
AGAINST ('+java +(linguaggio parole)' in BOOLEAN
                                          MODE);
```

cerca di campi che devono contenere il termine java e uno tra linguaggio o parole.

MODALITÀ DI ESPANSIONE DELLE RICERCHE

Questa modalità si attiva specificando "WITH QUERY EXPANSION" dopo i termini ricercati. Una espansione della query consiste nell'eseguire la query due volte: nella prima avviene una normale ricerca full text.
Da essa vengono reperiti i record con score maggiore di zero e, inoltre, vengono reperiti i termini più comunemente riscontrati. Questi altri termini vengono usati per una seconda query e anche i risultati di quest'ultima query sono riportati come risultato complessivo. Ecco un esempio:

```
SELECT *, MATCH (titolo, contenuto)
 AGAINST ('java' WITH QUERY EXPANSION)
  FROM pensieri_parole;
```

Osservando gli score si può notare che anche i record che non contengono "Java" ma "linguaggio" hanno uno score maggiore di zero e che, pertanto, compaiono nei risultati della ricerca full text con l'espansione dei termini.

ATTENZIONE AI CASI PARTICOLARI

Se un termine compare in più del 50% dei record, esso viene automaticamente escluso dalla ricerca full text. Questo è, ovviamente, molto sensato per le basi dati di grosse dimensioni (si pensi che, normalmente, questo tipo di termini sono articoli o aggettivi di uso comune). Però in basi dati specialistiche (per non dire monotematiche o quasi!) può rappresentare un problema di cui è opportuno tener conto. Molto più spesso è un problema solo nel caso di semplici test (come i nostri) dove i record sono davvero pochi e che, inizialmente, possono presentare una piccola difficoltà nel comprendere i risultati quando questo meccanismo entra in gioco!

LE NOVITÀ IN MYSQL 5.1

MySql 5.1 offre delle nuove API che permettono la creazione di plug-in aggiuntivi per la creazione di parser personalizzati per estendere (o sostituire!) le ricerche full text predefinite. Un esempio potrebbe essere quello di creare un plug in che automaticamente espande le ricerche fatte per un nome di rivista (come IoProgrammo) aggiungendo i nomi delle riviste dello stesso editore (Internet Magazine, per esempio) o l'editore stesso (Edizioni Master) senza che necessariamente tali informazioni siano inserite nei record (si usa cioè una meta-conoscenza che dà una gerarchia e/o relazione su alcuni termini). Per realizzare un siffatto plugin si può far riferimento al manuale utente fornito con MySql 5.1 e disponibile per il dowload alla pagina http://dev.mysql.com/doc/refman/5.1/en/.

RICERCHE FULL TEXT IN PHP E JSP

L'uso delle query full text ha enorme rilevanza sul Web; gli esempi allegati all'articolo mostrano due semplici applicazioni che eseguono le query inserite dall'utente in una form HTML. La prima applicazione (fullTextPhp) è scritta in PHP mentre la seconda (fullTextJsp) utilizza la tecnologia J2EE e, in particolare, le pagine JSP (**Figura 2**).
Per far funzionare l'esempio con le JSP è necessario eseguire il download del driver Connector/J dalla pagina http://dev.mysql.com/downloads/connector/j/

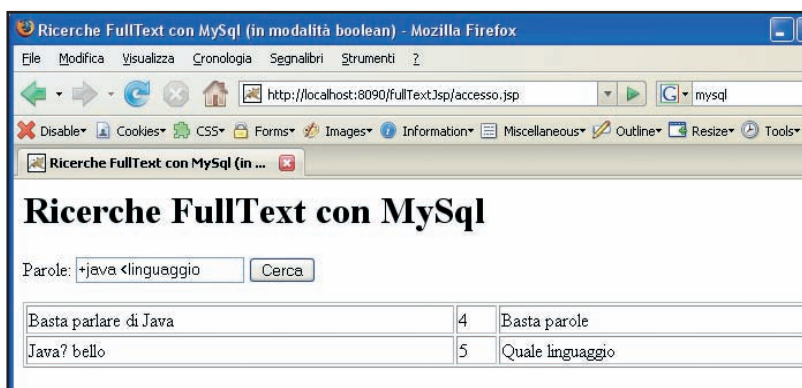
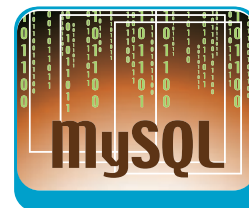
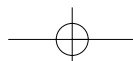
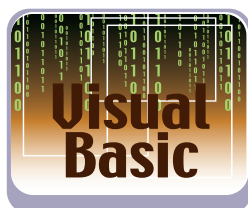



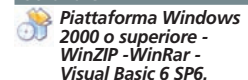
Fig. 2: ***La webapp realizzata con le JSP.***

5.0.html (una volta esploso lo ZIP, è necessario copiare il file JAR sotto la cartella common/lib del Tomcat, o una cartella analoga del Servlet Container usato!).
Come si potrà osservare, in entrambi i casi non sono da adottarsi particolari accorgimenti per sfruttare le ricerche full text: basterà far sì che la sintassi SQL delle query sia quella specifica per MySql. Purtroppo queste query sono specifiche per tale DBMS; d'altro canto le implementazioni tra DBMS diversi sono così poco omogenee che è impossibile riuscire ad unificare sintassi e caratteristiche di ricerca.

Ivan Venuti

GESTIRE FILE ZIP, RAR E CABINET CON VB

OPERARE SU FILE COMPRESSI PUÒ RISULTARE UTILE IN TUTTA UNA SERIE DI CIRCOSTANZE. TUTTAVIA, QUALI SONO GLI STRUMENTI CHE VISUAL BASIC CI METTE A DISPOSIZIONE PER LA CREAZIONE, LA MODIFICA, LA CONSULTAZIONE?

Tra i formati di compressione più comuni, del mondo Windows, sicuramente includiamo: ZIP, RAR e Cabinet. I primi due, di solito, sono gestiti con le applicazioni WinZip e WinRar; invece, i file Cabinet rappresentano lo standard di compressione per i file dei sistemi operativi di casa Microsoft. Essi sono archiviati con estensione CAB e al loro interno possono contenere più file compressi. I file CAB di solito vengono utilizzati per distribuire i pacchetti di installazione dei sistemi operativi, dei controlli Activex e delle Applicazioni. A tal proposito ricordiamo che in Visual Studio è presente l'applicazione creazione guidata pacchetti di installazione che, velocemente ed intuitivamente, permette di creare Pacchetti per distribuire le applicazioni Visual Basic. I file CAB possono essere manipolati anche con altri strumenti come le funzioni API, i Tool MakeCab ed Extract e gli stessi WinZIP e WinRAR.
Nell'articolo dopo aver fatto una breve descrizione delle tecniche utilizzate per interagire con i compressori di file, presentiamo un'applicazione che permette di creare file compressi nei tre formati citati. Inoltre vedremo come crittografare un file RAR. Per implementare alcuni degli esempi, dovete installare WinZip e WinRar e conoscerne le caratteristiche principali. Naturalmente potete installare la versione Shareware che trovate sui rispettivi siti web: www.winzip.com e www.winrar.com.

REQUISITI

Conoscenze richieste
Conoscenze di base sulla gestione dei progetti e dei controlli di Visual Basic.

Software
Piattaforma Windows 2000 o superiore - WinZIP -WinRar - Visual Basic 6 SP6.

Impegno

Tempo di realizzazione

CABINET, MAKECAB ED EXTRACT

Come accennato per distribuire un controllo Activex o un'applicazione Visual Basic è necessario creare almeno un file Cabinet con il Wizard: creazione guidata pacchetti d'installazione. In Windows

Fig. 1: ***Il primo step del Wizard creazione pacchetti di installazione.***

2000 e versioni superiori, però, i file CAB possono essere creati da programma con i Tool MakeCab.exe ed Extract.exe. Quando si deve creare un file CAB con MakeCab, innanzitutto, bisogna prevedere la creazione di un file di direttive con estensione DDF (acronimo che significa Diamond Directive File) che serve per dirigere MakeCab durante la creazione del file. Per capire come opera MakeCab consideriamo il seguente esempio di file DDF. Le principali direttive DDF sono schematizzare nella **Tabella 1**.

```
.Option Explicit
.Set Cabinet=on
.set Compress=on
.Set MaxDiskSize=CDRom
.set ReservePerCabinetSize=6144
.Set DiskDirectoryTemplate=
.Set CompressionType=MSZip
.Set CompressionLevel=7
.Set CompressionMemory=21
.Set CabinetNameTemplate=dati.cab
; questo è il file da comprimere
"C:\prova.avi"
```

Le direttive precedenti guidano MakeCab nella

compressione del file c:\prova.avi, esse vanno inserite in un file del blocco note salvato con estensione dff. Il file così ottenuto va passato a MakeCab come spiegheremo tra poco. In realtà con la nostra applicazione, tutti questi passaggi sono fatti dal programma.
Il *Tool MakeCab*, come avete intuito, quando è invocato riceve una serie di parametri, il significato di alcuni di essi è schematizzato nella **Tabella 2**. La sintassi completa per invocare MakeCab, invece, è la seguente.

```
MAKECAB [/Vn] [/D variable=value ...]
        [/L directory] source [destination]
MAKECAB [/Vn] [/D variable=value ]
        /F directives_file [...]
```

Nei nostri esempi utilizziamo il seguente comando

```
MAKECAB /F "Nomefile.ddf"
```

Per quanto riguarda il *Tool Extract*, che permette di estrarre dati dai file *Cabinet*, la sintassi del è la seguente:

```
EXTRACT [/y] [/A] [/D | /E] [/L location]
        cabinet_file [file_spec ...]
EXTRACT [/y] compressed_file [destination_file]
```

La prima forma viene utilizzata in generale, la seconda nel caso si voglia estrarre un solo file. I parametri principali del comando sono descritti nella **Tabella 3**.
Nei nostri esempi utilizziamo il seguente comando:

```
EXTRACT FileCab /e /l DirectoryEstrazione
```

Notate che *DirectoryEstrazione* è il nome della *directory* dove verranno posti i file estratti.
Per utilizzare *MakeCab* e *Extract*, strumenti esterni a *Visual Basic*, si può ricorrere alla funzione Shell. Questa però in alcuni casi presenta dei problemi con i nomi lunghi dei file (quelli non MsDos) per questo con *Shell* conviene utilizzare la funzione dell'*API GetShortPathName*, che consente di recuperare il path corto dei file. La sintassi di *Shell* la trovate in un Box laterale, la sintassi di *GetShortPathName*, invece, è la seguente:

```
Public Declare Function GetShortPathName Lib
        "kernel32" _
Alias "GetShortPathNameA" (ByVal lpszLongPath As
        String, _
ByVal lpszShortPath As String, ByVal cchBuffer As
        Long) As Long
```

Il primo parametro è la stringa che contiene il percorso lungo, il secondo è una stringa buffer dove verrà memorizzato il corrispondente percorso *DOS*, l'ultimo parametro è un numero che specifica la grandezza in *byte* del *buffer*, comprende anche il *Null* finale.
Per gestire i file *CAB*, in realtà, è anche possibile ricorrere ad alcune funzioni API che però non sono semplice da programmare dato che si poggiano su complicate funzioni di Callback.

Direttiva	Descrizione
.Option Explicit	Richiesta per la definizione delle variabili
.Set variable=[value]	Serve per impostare il valore di una Variabile
;	Commento all'interno del file DDF
Cabinet=ON | OFF	Imposta su on (generazione file CAB) oppure su off la modalità CAB
CabinetNamen=filename	Nome del file Cabinet
Compress=ON | OFF	Imposta su on oppure su off la compressione
CompressionLevel	Imposta il livello di compressione
CompressionMemory	Imposta la quantità di memoria utilizzata nel processo di creazione del file
CompressionType=MSZIP	Stabilisce il tipo di compressione da fare
DiskDirectoryTemplate	Indica il nome della Directory su disco
MaxDiskSize=CdRom	Con questa impostazione non si fissano dimensioni massime per lo spazio disco occupato
ReversePerCabinetSize=6144	Riserva Spazio per firma digitale

Tabella 1 Direttive DDF

Parametro	Descrizione
Source	File che deve essere compresso
Destination	Nome del file di destinazione
/D variable=value	Per impostare una variabile
/L directory	Specifica la directory di destinazione del file compresso
/F directives_file	File che contiene le direttive di compressione, il nostro file ddf.

Tabella 2 Parametri di MakeCab

Parametro	Descrizione
/A	Per processare tutti i file CAB
/D	Restituisce solo la lista delle directory senza estrarre i file.
/E	Forza l'estrazione dei file.
/L	Specifica la directory di estrazione.
/Y	Sovrascrive i file estratti senza chiedere conferma.
Compressed_file	È utilizzato per l'estrazione di un singolo file che verrà trasferito nella directory di destinazione.
Destination_file	Percorso della directory di destinazione.
Cabinet_file	Nome del file CAB.
Location	Percorso della directory in cui si trova il file da estrarre.
File_spec	Specifica il tipo di file che si vuole estrarre da un file CAB, ad esempio con *.DOC estraiamo tutti i file con estensione DOC

Tabella 3 Parametri di Extract

WINZIP E WINRAR

WinZip è il più diffuso software di compressione

per *Windows*. *Zip* infatti è sinonimo di file compresso. Negli ultimi anni però accanto a *WinZip* si sono affermati diversi programmi di compressione, tra i quali *WinRar*, e la stessa *Microsoft* ha inserito le funzionalità di compressione nei propri sistemi operativi. Tra le caratteristiche di *WinZip* ricordiamo: gli algoritmi di crittografia, per proteggere i dati sensibili, con il supporta della *crittografia AES* a *128* e *256 bit*; la possibilità di salvare i file zippati direttamente su *CD* o *DVD*, il supporto per processori a 64 bit ecc. Inoltre fornisce vari *Wizard* che permettono di ottimizzare le principali operazioni. Con *WinZip* è possibile aprire anche file *BZ2*, *RAR* e *CAB*. I vantaggi di *WinZip* li possiamo sintetizzare in popolarità, semplicità e velocità. Per quanto riguarda *WinRar*, invece, si tratta di uno strumento utile soprattutto per effettuare Backup infatti nell'ultima versione è stato migliorato ulteriormente l'algoritmo di compressione per funzionare al meglio con i nuovi processori *Dual Core*. *WinRar* permette di creare e estrarre archivi nei formati più comuni tra i quali *ZIP, CAB, ARJ, LZH, TAR, GZ e TAR.GZ*. Il formato *RAR*, inoltre, consente una migliore compressione rispetto allo *ZIP* e permette: la creazione e la gestione di archivi multivolume, il recupero di archivi danneggiati, il blocco dell'archivio per evitare modifiche accidentali ed è in grado di gestire file più grandi di 8,5 G byte.

Per utilizzare *WinZip* e *WinRar* da *Visual Basic* possiamo impostare delle particolari righe di comando da sottoporre alla funzione *Shell*. Per *WinZip* il comando può essere impostato secondo la seguente sintassi:

```
WINZIP32.EXE [-min -a -u -p -s ...] [options]
                                filename[.zip] files
```

Dove i parametri hanno il seguente significato: min rende minimo l'aggiunta del file, -a indica che si aggiungono dei file, -u indica che si sta facendo un update, -p specifica che si vogliono archiviare i *path* delle directory che contengono i file. Se invece si vuole crittografare l'archivio si può usare il parametro *-sPassword*, dove *Password* è un stringa *case-sensitive*, ecc. Poi *filename* è il nome del file compresso, *files* sono i nomi dei *files* da comprimere. Per estrarre i file *Zippati*, invece, utilizzeremo una istruzione come la seguente:

```
WINZIP32.EXE -e [-o -s] filename[.zip]  folder
```

Vediamo le varie parti: *-e* è sempre richiesto, *-o* per sovrascrivere i file esistenti senza mostrare un prompt di conferma, *-s* per specificare la *Password* di estrazione. *Filename* è il nome del file compresso mentre *folder* è la directory dove verranno inseriti i file estratti.

Per *WinRar* la sintassi completa del comando è la seguente:

```
WinRAR  <comando>  -<opzione>  -<opzioneN>
                  <archivio>  <file...>  <@lista_file...>
                              <percorso_d'estrazione>
```

Comando è composto da uno o più caratteri che determinano la funzione eseguita da *WinRAR*, *opzione… opzioneN* sono le opzioni utilizzate per specificare determinati tipi di operazioni: livello di compressione, tipo di archivio, ecc. In particolare nei nostri esempi per comprimere i file utilizziamo un'istruzione come la seguente:

```
WinRAR.exe a –p[PassWord] nomefile.rar @ files
```

Dove il comando a permette di aggiungere un file all'archivio compresso, *-p* seguito dalla password permette di specificare una *password* per l'archivio compresso, *nomefile.rar* è il nome e il percorso del file *RAR* creato, infine @ è seguito dal nome e dal percorso dei file che devono essere compressi. Per esempio con il seguente si crea un file *RAR* di nome *prova.rar* protetto dalla *Password* ioprogrammo

```
WinRAR  a  -pioprogrammo  prova  c:/topsecret.txt
```

Per estrarre il contenuto di un archivio invece utilizzeremo la seguente.

```
WinRAR.exe e -ppassword path+nomefile.rar
                                  path+dirprova
```

In questo caso si estraggono i file dell'archivio compresso, che ha una *password*, e si spostano in una directory di nome *dirprova*.

ALZIP E ZIP GENIUS

Navigando su internet ci s'imbatte in una marea di programmi per la creazione di archivi compressi, tra questi segnaliamo AlZip (www.altools.net) e ZipGenius (www.zipgenius.it/ita/). AlZip permette di gestire 8 formati di archivi compressi e visualizzarne almeno fino a trentasei. Inoltre consente di utilizzare Password che possono essere recuperati nel caso di dimenticanza. ZipGenius invece è il più conveniente, dato che è freeware, semplice da utilizzare e perfettamente integrato con Windows. Supporta oltre a 20 formati di archivi compressi, tra i quali le immagini di CD-ROM e DVD in formato ISO9660. Inoltre è perfettamente compatibile con OpenOffice e il formato 7-Zip (www.7-zip.org).

APPLICAZIONE PER LA MULTI-COMPRESSIONE

L'applicazione che presentiamo gestisce archivi nei formati *ZIP*, *CAB* e *RAR*. Essa permette di ricercare e raggruppare i file da comprimere e ricercare il file

compresso da decomprimere. Di seguito per punti descriviamo come implementare il progetto.

1 Create un progetto *EXE*, che referenzia la *MS Common Dialog*, con una *Form* e un modulo *Bas* di supporto. Sulla *Form* prevedete i componenti per ricercare i *file*, raggrupparli e invocare i *Tool*: *WinZip*, *WinRar*, *MakeCab* ecc. A tal fine per ricercare i file prevedete un *DriveListBox*, un *DirListBox* e un *FileListBox*; mentre per raggrupparli prevedere una *ListBox*. Per invocare i Tool basta prevedere sei pulsanti, cioè 3 coppie Crea-Estrai, nominati: *CreaZip*, *CreaRAR*, *CreaCAB*, *EstraiZip*, *EstraiRAR* e *EstraiCAB*. Inoltre per ricercare i file compressi prevedete: un pulsante (nominato *CmdEx*), un *textbox* (nominato *TextDDF*) e un controllo *Common Dialog*. Infine solo per i file *RAR* prevedete un *TextBox* (denominato *txtpw*) per specificare la *password* di compressione/ decompressione. Controllate la figura 3.

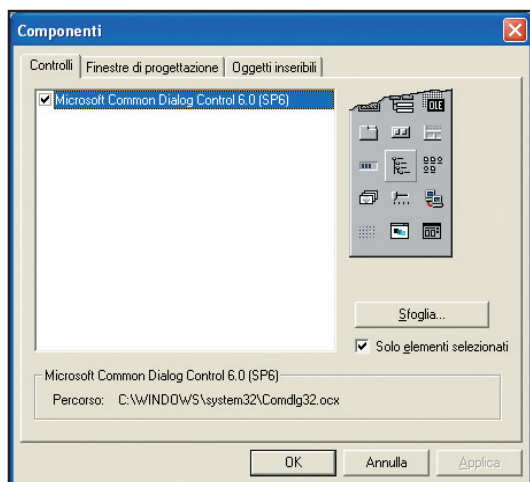



Fig. 2: *I componenti referenziati dal progetto*

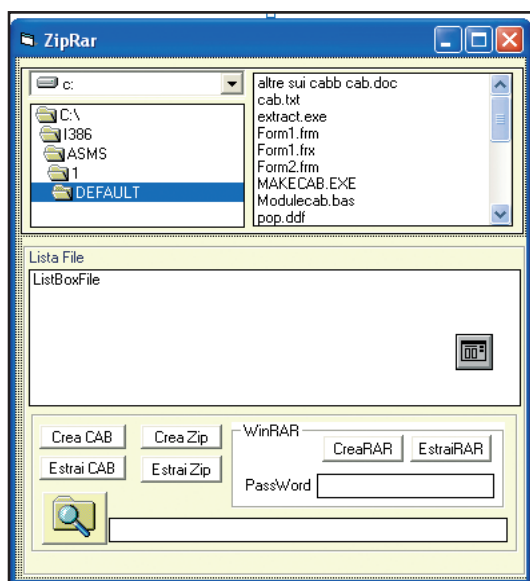



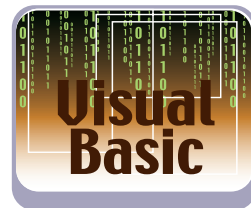
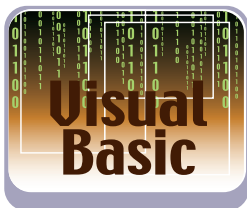
Fig. 3: *La form in fase di progettazione.*

2 Il codice per la ricerca dei file, da inserire nel DriveListBox, nel DirListBox e nel FileListBox, è il seguente.

```
Private Sub dirList_Change()
'DirListBox
   filList.path = dirList.path
End Sub
Private Sub DirList_LostFocus()
   dirList.path = dirList.List(dirList.ListIndex)
End Sub
Private Sub DrvList_Change()
'DriveListBox
   On Error GoTo DriveHandler
   dirList.path = drvList.Drive
   Exit Sub
DriveHandler:
   drvList.Drive = dirList.path
   Exit Sub
End Sub
Private Sub filList_DblClick()
'FileListBox
 Dim i As Integer
 For i = 0 To filList.ListCount - 1
  If filList.Selected(i) = True Then
   ListBoxFile.AddItem dirList.path & "\" &
                                    filList.List(i)
  End If
 Next i
End Sub
Private Sub ListBoxFile_DblClick()
 For i = 1 To ListBoxFile.ListCount
  If ListBoxFile.Selected(i - 1) = True Then
   ListBoxFile.RemoveItem i - 1
   Exit Sub
  End If
 Next
End Sub
```

S'intuisce che i *file* da comprimere sono raggruppati nella *ListBox1*. Nella *ListBoxFile_DblClick*, invece, abbiamo inserito il codice per cancellare gli elementi dalla *ListBox*. Così basta un doppio *clic* per eliminare un file dal raggruppamento.

3 Nel modulo *Bas* di supporto vanno inserite le dichiarazioni di alcune costanti e delle funzioni che permettono di valutare il *path* corto di un *file*.

```
Public Const pathzip =
                "C:\programmi\winzip\WinZip32.exe"
Public Const pathrar =
                 "C:\programmi\winrar\WinRAR.exe"
Public path As String
'contiene il path corto del progetto
Public Declare Function GetShortPathName Lib
                                    "kernel32" _
```

```
Alias "GetShortPathNameA" (ByVal lpszLongPath As String, _
ByVal lpszShortPath As String, ByVal cchBuffer As Long) As Long
Public Function pathnomecorto(ByVal sLongFileName As String) As String
   Dim lRetVal As Long, sShortPathName As String, iLen As Integer
   sShortPathName = Space(1200)
   iLen = Len(sShortPathName)
   lRetVal = GetShortPathName(sLongFileName, sShortPathName, iLen)
   pathnomecorto = Left(sShortPathName, lRetVal)
End Function
```

La funzione *pathnomecorto* è invocata ogni volta che si deve usare il percorso *MSDos* di un file. In particolare nella *Form_Load* impostiamo il percorso corto della directory che contiene il progetto.

```
Private Sub Form_Load()
 path = pathnomecorto(App.path + "\")
End Sub
```

4 Dopo aver ricercato e raggruppato i file, per avviare la creazione dell'archivio *CAB* utilizziamo il codice seguente.

```
Private Sub CreaCAB_Click()
 Open App.path + "\" + Me.TextDDF + ".ddf" For Output As #1
   Print #1, ".Option EXPLICIT"
   Print #1, ".Set Cabinet=on"
   Print #1, ".Set Compress=on"
   Print #1, ".Set MaxDiskSize = CDROM"
   Print #1, ".Set ReservePerCabinetSize = 6144"
   Print #1, ".Set DiskDirectoryTemplate ="
   Print #1, ".Set CompressionType = MSZIP"
   Print #1, ".Set CompressionLevel = 7"
   Print #1, ".Set CompressionMemory = 21"
   Print #1, ".Set CabinetNameTemplate=" + TextDDF + ".cab"
   For i = 1 To ListBoxFile.ListCount
    Print #1, pathnomecorto(ListBoxFile.List(i - 1))
   Next
  Close #1
  Dim RetVal As Integer
  RetVal = Shell(path + "makecab.exe" + " /f " + path + TextDDF _
  & ".ddf", vbMaximizedFocus)
 End Sub
```

Con la *CreaCAB*, prima è creato il file delle direttive, poi viene invocato il *Tool makecab.exe*. Il nome del *file dff* è specificato in *TextDDF*, questo sarà anche il nome del file *CAB* creato. Notate che sono salvati nella *directory* del progetto.
Per estrarre i file da un archivio *CAB*, invece, utilizziamo il seguente codice.

```
Private Sub EstraiCAB_Click()
  Shell path + "extract.exe " + pathnomecorto(TextDDF) + " /e /l " _
  & path + "DirExt", vbMaximizedFocus
 End Sub
```

Notate che nelle procedure precedenti ipotizziamo che i *Tools* si trovano nella *directory* del progetto e che esista la *directory DirExt* (dove sono decompressi i file *Cab*).

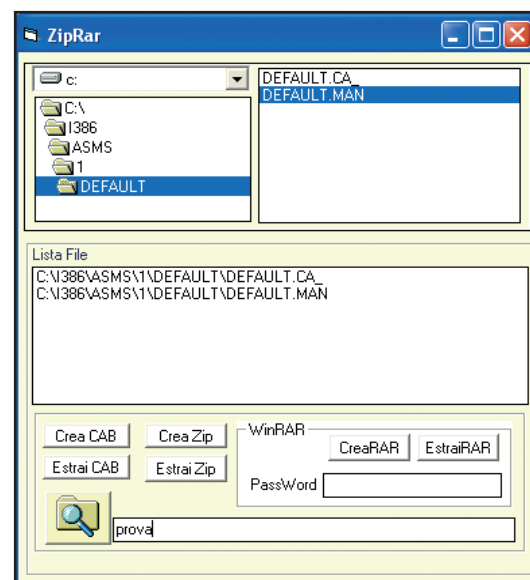



Fig. 4: ***La form in fase di esecuzione.***

5 Per gestire gli archivi *RAR* utilizziamo il seguente codice

```
Private Sub CreaRAR_Click()
 On Error GoTo errore
 Dim password As String
 Dim comando As String
 Dim files  As String
 If txtpw <> "" Then
```

CONTROLLI ACTIVEX PER INTERNET

I controlli Activex per internet sono distribuiti con file Cabinet prodotti, per esempio, con il Wizard creazione guidata pacchetti di installazione. Con questo Wizard, per creare un file Cabinet per Internet, al secondo passaggio della procedura, bisogna selezionare Internet Package. Tra i vari passaggi del Wizard, poi, c'è quello in cui si deve specificare se il controllo è sicuro per lo scripting e l'inizializzazione. Questo serve a certificare che l'Activex non creerà problemi di nessuna natura sul computer Client che l'utilizzerà. Ricordiamo che per inserire un controllo Activex in una pagina HTML bisogna utilizzare i Tag <Object> e <Param>.

```
password = "-p" + txtpw
Else
password = ""
End If
For i = 1 To ListBoxFile.ListCount
files = files + " " +
        pathnomecorto(ListBoxFile.List(i - 1))
Next
comando = pathnomecorto(pathrar) & " a " &
        password _
& " " & path + "dirwinrar\" + TextDDF & ".rar @"
Shell comando

Exit Sub

errore:
MsgBox "Si è verificato il seguente errore:" _
& Err.Description, vbInformation, App.Title
End Sub
Private Sub EstraiRAR_Click()
On Error GoTo errore
Dim password As String
Dim comando As String
If txtpw <> "" Then
password = "-p" + txtpw
Else
password = ""
End If
comando = pathnomecorto(pathrar) & " e " &
        password _
& " " & pathnomecorto(TextDDF) & " " & path +
        "DirWinRar\"
Shell comando
Exit Sub
errore:
MsgBox "Si è verificato il seguente errore:" _
& Err.Description, vbInformation, App.Title
End Sub
```

Notate l'utilizzo del parametro *password* nella creazione del comando di compressione/decompressione. Anche in questo caso s'ipotizza che esista la *DirWinRar* e che *WinRar* sia installato nella directory specificata in *PathRar*.

6 Per gestione i file *ZIP* utilizziamo il seguente codice.

```
Private Sub CreaZip_Click()
On Error GoTo errore
Dim comando As String
Dim files  As String
For i = 1 To ListBoxFile.ListCount
files = files + " " +
        pathnomecorto(ListBoxFile.List(i - 1))
Next
comando = pathnomecorto(pathzip) & " -min -a " _
& path + "\dirzip\" + TextDDF & ".zip " & files
Shell comando
Exit Sub
errore:
MsgBox "Si è verificato il seguente errore:" _
& Err.Description, vbInformation, App.Title
End Sub
Private Sub EstraiZIP_Click()
Dim comando As String
On Error GoTo errore
comando = pathnomecorto("pathzip") _
& " -e -o " & pathnomecorto(TextDDF) & " " & path
        + "\dirzip\"
Shell comando
Exit Sub
errore:
MsgBox "Si è verificato il seguente errore:" _
& Err.Description, vbInformation, App.Title
End Sub
```

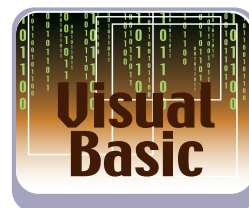
7 Per ricercare un file compresso invece utilizziamo il seguente codice.

```
Private Sub CmdEx_Click()
CommonDialog1.FileName = ""
OpenFile
End Sub
Sub OpenFile()
If CommonDialog1.FileName = "" Then
With CommonDialog1
.Filter = "tutti (zip,rar,cab)|*.zip;*.rar;*.cab|" _
& "Cabinet (*.CAB)|*.CAB |winZIP
        (*.ZIP)|*.ZIP;*.zip; |" _
& "winRAR (*.RAR)|*.rar;*.RAR "
.ShowOpen
End With
End If
If CommonDialog1.FileName <> "" Then
TextDDF = CommonDialog1.FileName
End If
End Sub
```

In conclusione facciamo notare che nel codice proposto manca la gestione degli errori e i controlli sul tipo di operazione eseguita.

CONCLUSIONI

Nel corso dell'articolo abbiamo introdotto alcune tecniche per la creazione di file compressi. Il progetto d'esempio può essere considerato la base di partenza per l'approfondimento degli argomenti, a tal fine vi consigliamo di leggere le parti di Help di WinRar e WinZIP che trattano di command line.

Massimo Autiero

ELABORARE REPORT LATO SERVER

VEDIAMO COME NON FAR GRAVARE SUI CLIENT L'ONERE DI ELABORAZIONE E DI MEMORIA RICHIESTI PER GENERARE COMPLESSI REPORT, SEMPLICEMENTE SPOSTANDO QUESTO PROCESSO SU UN SERVER CONDIVISO PROGETTATO DA NOI

Lo sviluppo di effervescenti architetture lato server e di nuovi modi per scrivere le applicazioni ha, se vogliamo, riportato indietro l'orologio a quando anni fa, esisteva un sistema host centralizzato (AS/400, i vari mainframe e i server unix) e le applicazioini giravano semplicemente in terminali stupidi a costo di CPU zero per i client. Poi, prima i PC e poi i PC con Windows hanno dimostrato che forse le macchine client potevano metterci del loro, ad esempio per offrire ricche interfacce grafiche inimmaginabili per i vecchi host.
Sembrava che fosse una china non destinata ad essere cambiata fino a quando è arrivato il web e le cose sono tornate al punto di partenza e il client si limita a ricevere e mostrare, senza elaborare nulla. Per le applicazioni gestionali tradizionali desktop questo cambiamento non è ancora arrivato del tutto, ma qualcosa è stato già mutuato (i database server) e si comincia ad intravvedere uno sviluppo di tipo distribuito, cioè con un application server che sgravi i client di parte del carico. Ma in attesa che questo approccio si faccia pervasivo, perché intanto non cominciare a spostare qualcosa sul server?
Vediamo... cos'è quella cosa tanto bella da far vedere ai clienti, tanto semplice (nella maggior parte dei casi) da scrivere ma tanto pesante da far girare sulle macchine dei nostri clienti, sempre troppo lente per i nostri gusti e sempre pieni di ammennicoli che le rendono ancora più lenti? Ma certo, i report! Esistono in commercio soluzioni centralizzate, ad esempio Crystal Reports Server, ma o sono troppo costose per i ridotti budget ormai a disposizioni per i software gestionali (Crystal Reports Server XI costa 6000 euro per solo 5 accessi client concorrenti), o sono poco integrabili con le nostre soluzioni. Spesso richiedono un impegno notevole in fase di progettazione e deployment e si limitano a sparare i report via browser solo in formato html (e quindi con qualità non eccellente rispetto ai formati proprietari o PDF) o in buona qualità ma via browser e quindi non fruibili all'interno delle applicazioni, come il nostro utente tipo è abituato a fare.
E allora cerchiamo di porre rimedio al problema realizzando in casa una soluzione semplice ed economica, ma non per questo meno efficace.

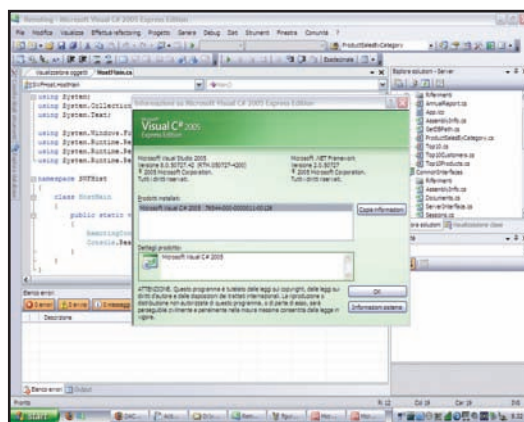



Fig. 1: *Visual C# 2005 Express in tutto il suo splendore*

LA LISTA DELLA SPESA

Siccome vogliamo realizzare un'applicazione desktop C#, possiamo certamente sviluppare l'intera soluzione con Microsoft Visual C# Express 2005 che è completamente gratuito.
Per la reportistica, invece, tra i vari prodotti presenti sul mercato per la piattaforma .NET, è certamente degno di nota ActiveReports .NET, giunto ormai alla versione 3.0 e prodotto da Datadynamics. Si tratta di un prodotto .NET nativo, a differenza soluzioni ibride che prevedono un layer .NET di compatibilità ed un motore, invece, basato su codice compilato unmanaged. Dal punto di vista delle caratteristiche di reportistica pura non spicca particolarmente per funzionalità, lo si può far rientrare nella media di questi prodotti, ma dal punto di vista dello sviluppatore di soluzioni integrate offre sicuramente degli spunti interessanti, come vedremo nel proseguo. Dal sito è possibile scaricare la versione demo, valida per sem-

pre, ma che produce un fastidioso watermark al piede del report.
Il costo del prodotto, nella versione standard, è di circa 500 euro, quindi allineato con la concorrenza. Una volta scaricato il prodotto, si installa come addin di Visual C# Express e sembra funziona-

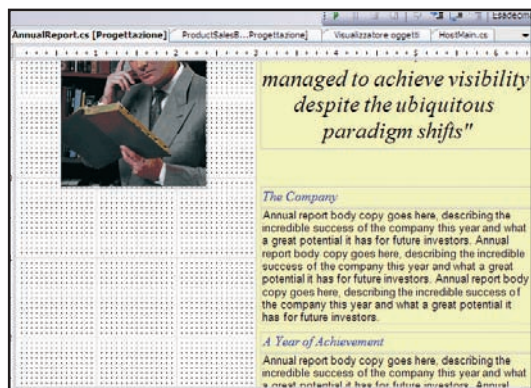



Fig. 2: ***Il designer di ActiveReports .NET 3.0 all'interno di Visual C# 2005 Express***

re correttamente e senza limitazioni anche all'interno della versione gratuita dell'ambiente di sviluppo Microsoft. Offre un potente designer grafico di report e tutto ciò che occorre per progettare e scrivere reportistica in applicaizioni moderne.
Il terzo ingrediente speciale di questa torta è la tecnologia di remotizzazione .NET Remoting, che consente la chiamata di servizi remoti in modo da realizzare applicazioni distribuite e quindi non semplicemente client-server, ma con un client e un modulo server di tipo applicativo (un application server) su cui far girare parte delle nostre applicazioni, magari le parti più pesanti sul server del client, lasciando liberi i client spesso più precari in termini di risorse hardware. Ed è proprio questo che andremo a fare, ed è proprio sul nostro host-server che girerà il report, verrà completamente elaborato facendo i suoi dovuti accessi al database e, infine, trasmetterà al client soltanto un flusso corrispondente al report già elaborato ed impaginato, pronto per essere mostrato in anteprima nel client per le successive stampe. .NET Remoting è stato già oggetto di due articoli apparsi su ioProgrammo e visto lo studio di questo protocollo non sarà oggetto di questo articolo, si rimanda agli articoli in questione, alla numerosa documentazione sull'argomento disponibile su internet, ai volumi sull'argomento e, per i più pigri, al Riquadro Microsoft .NET Remoting, che certo non vi farà diventare esperti sulla materia, ma almeno è un inizio. Ad ogni modo, il codice fornito col presente articolo è funzionante, auto-consistente e quindi certamente un buon punto di partenza, oltre che per lo studio della soluzione di reportistica presentata, anche per la tecnologia .NET Remoting.

ACTIVEREPORTS .NET E LE SUE SEGRETE DOTI

A differenza di altre soluzioni commerciali ActiveReports, non prevede una funzionalità server e cioè di funzionare al di fuori dell'applicazione che ospita il report stesso, ma offre delle interessanti caratteristiche che si offrono alla remotizzazione. E quindi il resto ce lo mettiamo noi... Ma andiamo con ordine ed osserviamo un po' meglio ActiveReports .NET, attraverso uno degli esempi offerti dal setup stesso del prodotto e richiamabili dalla voce *Programmi --> Datadynamics --> ActiveReports for .NET 3.0 --> Samples --> CSharp*. Richiamiamo l'esempio AnnualReport che, dopo qualche minuto necessario alla conversione del progetto dalla versione 2003 di Visual Studio .NET usata per produrlo, alla nostra corrente versione 2005, ci conduce ad un tipo progetto WinForms contenente al suo interno una serie di classi-report (AnnualReport.cs, Top10.cs, ecc..). Tra questi file troviamo però un form nel file StartupForm.cs. Esso ospita il controllo di Report Preview offerto da ActiveReports che serve a mostrare il report AnnualReport elaborato. Osserviamo il blocco di istruzioni che permette di istanziare il report, elaborarlo, agganciarlo al controllo di preview e di mostrare a video il risultato attraverso il form custom dell'applicazione:

```
// BLOCCO 1
// Setup a new instance of the AnnualReport
AnnualReport rpt = new AnnualReport();
//Run the report, and set it to the viewer control on
                                              the form
rpt.Run();
// BLOCCO 2
this.arvMain.Document = rpt.Document;
```

A parte il termine blocco, sono riportati i commenti originali del codice di esempio, ma in pratica si tratta semplicemente di istanziare il report (la classe *AnnualReport*), elaborarlo attraverso il metodo *Run()* e assegnare il risultato dell'elaborazione (la proprietà Document del report elaborato) all'oggetto di preview ospitato nel form. A quel punto è sufficiente invocare la *.Show()* del form WinForms per vedere il nostro report in bella mostra. Questo è l'approccio classico di ActiveReports .NET e delle altre decine di prodotti di reportistica esistenti sul mercato, non solo per la piattaforma .NET, ma il nostro obiettivo sarebbe quello di far girare la parte di codice segnata come BLOCCO 1 sul server, inviare il risultato al client e qui eseguire il BLOCCO 2. Detto così, a parte l'idea innovativa in se, sembrerebbe facile, ma purtroppo non è così banale perché gli oggetti della libreria ActiveReports

ETL

Il processo di estrazione dei dati, extraction, è il processo usato per ottenere dati da database, archivi, file. Poiché i dati possono venire da sorgenti diverse, la trasformazione, transformation, è il processo per rendere coerenti tutti i dati. Il processo finale di caricamento, load, serve per inserire i dati nel sistema informatico finale.

.NET non sono stati progettati per questo scopo e, scendendo nei dettagli, i due oggetti rilevanti per la bisogna, e cioè *AnnualReport*, che discende da *DataDynamics.ActiveReports.ActiveReport3* e *Document*, classe direttamente ereditata da *System.Object*, non offrono le caratteristiche sufficienti alla remotizzazione. Infatti, se evidentemente Document ne è completamente come si evince dalla classe da cui deriva, ActiveReport3, invece, discenda da *MarshalByRefObject*. E qui si apre una parentesi interessante: in realtà questa classe è proprio la regina della remotizzazione perché permette di essere fruita da un *AppDomain* remoto (si rimanda alla documentazione specifica di Remoting per una migliore conprensione), ma questo approccio, cioè eseguire una classe che si trova in remoto, è ottimo per numerose circostanze, ma non si presta in questo caso perché presenterebbe grossi problemi prestazionali. Un report, infatti, può avere anche dimensioni ragguardevoli, una volta processato e generato e così l'approccio della fruizione remota non fa al caso nostro. Inoltre è curioso notare che, fino alla versione 2.0 di ActiveReports .NET, questo oggetto ereditasse direttamente da System.Object. Evidentemente questa variazione indica un'avvisaglia di tentativi di produrre un'architettura remotizzata.
Fortunatamente, con la soluzione che andiamo a proporre, saremo in grado di risolvere questi problemi e potremo utilizzarla anche con le versioni 1.0 e 2.0 del prodotto, qualora si fosse acquistata una di queste versioni e si volesse preservare l'investimento e/o i report già progettati. Ma veniamo finalmente al punto: come processare lato server un report e poi trasmettere al client solo il risultato, magari per mostrarlo in anteprima all'utente? Osserviamo questo interessante brano di codice commentato:

```
//istanziazione del report
AnnualReport rep = new AnnualReport();
//esecuzione del report
rep.Run(true);
//istanziazione di un MemoryStream
MemoryStream stream = new MemoryStream();
//salvataggio del risultato del report in uno stream
rep.Document.Save(stream);
```

In pratica l'oggetto Document di ActiveReports presenta il metodo Save e in partcolare la versione in overload che permette di persistere il risultato del report elaborato su uno Stream. Questa caratteristica è fantastica e fa proprio al caso nostro perché gli stream sono lo strumento più adeguato per la trasmissione di informazioni tra nodi remoti. Ed infatti, specularmente, è possibile deserializzare un report a partire da uno stream, come mostrato di seguito:

```
//recupera il viewer ActiveReports inserito nel form
                         di visualizzazione del report
DataDynamics.ActiveReports.Viewer.Viewer viewer1
                                      = this.viewer;
//deserializzazione del report a partire dallo stream
viewer1.Document.Load(stream);
//visualizzazione dell'anteprima del report
viewer1.Show();
```

È evidente che, disaccoppiando questo codice e tenendo la prima parte sul server e la seconda sul client, otteniamo esattamente il nostro scopo: un client leggero che si limita a mostrare un report già processato, quasi fosse un documento PDF già esistente, ed un server, magari ben corazzato, che elabora il report e ha accesso al base dati.

PROGETTIAMO UN SERVER DI REPORTISTICA

La tecnica è bella e pronta, ma adesso è il caso di mettere un po' d'ordine e di proporre una soluzione completa e pronta all'uso per sfruttare questa nuova interessante scoperta. Innanzitutto progettiamo una semplice interfaccia che dovrà essere implementata dal nostro server di reportistica:

```
namespace ioProgrammo.RemotingServices
{
   public delegate Stream ProcessReport
         (Guid sessionId, IDictionary parameters);
        public interface IRemoteServer
        {
        //LOGIN
        Guid Login(string username,
                                    string password);

               //INVOCA REPORT REMOTO
        Stream ProcessReport(Guid sessionId,
                              IDictionary parameters);
        }
}
```

Tale server sarà in grado di servire più richieste provenienti dallo stesso client o da client diversi, pertanto è necessario che ogni client si lasci identificare ad ogni richiesta. Ecco la ragione del metodo Login, che richiede gli immancabili username e password e restituisce un banale Guid che identifichi la sessione appena avviata in caso di autenticazione andata a buon fine. Le informazioni della sessione appena creata, e di tutte le altre già esistenti, saranno conservate dal no-

stro Report Server in un dizionario di classi Session, che è definita come mostrato di seguito:

```
namespace ioProgrammo.RemotingServices
{
    public class Session
    {
        public string Username = null;
        public Guid SessionId = Guid.Empty;
        public DateTime StartTime = DateTime.
                                        MinValue;
        public DateTime LastAccessTime = DateTime.
                                        MinValue;
    }
}
```

La classe presenta le informazioni identificativi essenziali della sessione e cioè lo username richiedente, il Guid assegnato al momento della Login, l'ora di creazione della sessione e l'ora dell'ultimo accesso, informazioni che evidentemente dovrà essere aggiornata ad ogni accesso al server da parte della sessione. Quest'ultima informazione può risultare molto utile qualora si volessero implementare meccanismi automatici di Session Timeout, cioè di decadimento della sessione dopo un certo periodo di inattività.
Ed ecco la pratica variabile Dictionary di tipo generic che conserverà tutte le sessioni correntemente attive e il metodo di recupero di tali informazioni:

```
private Dictionary<string, Session> _sessions =
                    new Dictionary<string, Session>();
private Session _getSession(Guid sessionId)
{
    foreach (Session session in _sessions.Values)
    {
        if (session.SessionId == sessionId) return
                                        session;
    }
    return null;
}
```

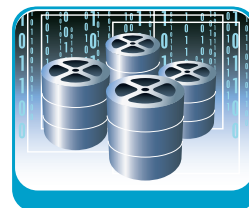
Osserviamo, infine, il metodo di login vero e proprio:

```
public Guid Login(string username, string password)
{
    Hashtable logins = new Hashtable();
    logins["ciro"] = "wow";
    logins["ciro1"] = "wow";
    logins["pinuccio"] = "wow3";
    //controllare username e password
    if (logins.ContainsKey(username.ToLower()) &&
            logins[username.ToLower()].ToString() ==
                                        password)
    {
```

```
        if (_sessions.ContainsKey(username
                                .ToLower()))
        {
            _sessions[username.ToLower()]
                        .LastAccessTime =
                            DateTime.Now;
            return _sessions[username.ToLower()]
                                .SessionId;
        }
        else
        {
            Session session = new Session();
            session.SessionId = Guid.NewGuid();
            session.Username = username;
            session.StartTime = DateTime.Now;
            session.LastAccessTime = DateTime.Now;
            _sessions[username.ToLower()] = session;
            return session.SessionId;
        }
    }
    return Guid.Empty;
}
```

Per semplificazione la verifica dello username e della password sono stati volutamente banalizzati. Un metodo certamente più interessante, già incontrato nell'interfaccia IRemoteServer ed implementato qui di seguito, è ProcesReport, che è proprio il cuore della nostra applicazione, che si occupa di processare lato server il report che poi verrà inviato al client richiedente in forma di stream, e questo, con la tecnica in precedenza esaminata, sarà in grado di deserializzarlo per mostrare il risultato in anteprima di stampa. Eccone il codice:

```
public System.IO.Stream ProcessReport
    (Guid sessionId, System.Collections.IDictionary
                                    parameters)
{
    Session session = _getSession(sessionId);
    if (session == null) return null;

    AnnualReport rep = new AnnualReport(this,
                                    parameters);
    rep.Run(true);
    MemoryStream stream = new MemoryStream();
    rep.Document.Save(stream);
        return stream;
}
```

Il metodo riceve come parametri l'id di sessione del client richiedente, informazione che viene immediatamente verificata a partire dal Dictionary generico che enumera sul server le sessioni aperte, e un secondo parametro Dictionary contenente i parametri da passare al report da elaborare, parametri evidentemente provenien-

NOTA

RIFERIMENTI

[1] Download gratuito di Microsoft Visual C# Express Edition - http://msdn.microsoft.com/vstudio/express/visualcsharp/default.aspx

[2] Download gratuito di Microsoft SQL Server 2005 Express Edition e di SQL Server - http://msdn.microsoft.com/vstudio/express/sql/default.aspx

[3] Download della versione demo di Datadynamics Active Report .NET 3.0 - http://www.datadynamics.com/Forums/72/ShowForum.aspx

ti dal client. Questi parametri servono alla corretta configurazione ed esecuzione del report e sono l'unico modo che ha il client per comunicare con il server e, in particolare, con il report che dovrà essere elaborato. Pertanto, se ad esempio l'utente, attraverso l'interfaccia utente del client, volesse scegliere di stampare una fattura in particolare o una lista di movimenti di magazzino da data a data o volesse semplicemente decorare il report con il logo aziendale anzichéno, tutte queste informazioni dovrebbero essere inserite nel dictionary, ciascuna con una sua specifica chiave di identificazione convenzionalmente riconosciuta dal report che dovrà essere invocato. Il server istanzia il report in questione passandogli il parametro parameters, magari al costruttore del report stesso, processa il report, serializza il risultato in un MemoryStream che infine restituisce come valore di ritorno del metodo ProcessReport. Nell'esempio il metodo implementativo ProccessReport istanzia esplicitamente il report AnnualReport, ma una eventua-

Fig. 5: : Avvio multiplo di progetti da Visual Studio 2005 Express

le implementazione davvero generica del nostro report server dovrebbe essere in grado di istanziare report diversi per far fronte a diverse esigenze dei client e tali report dovrebbero preferibilmente essere contenuti in assembly caricati dinamicamente al momento dell'istanziazione. Questa tecnica, possibile grazie alle potenti caratteristiche di reflection di .NET, è stata mostrata già in alcuni mie articoli apparsi su ioProgrammo.

Siamo così pronti ad inserire il nostro Report Server in un host in grado di renderlo disponibile come server di Remoting. Nel codice di esempio allegato, infatti, è stata definita la console application contenuta nel progetto Host.csproj che funge proprio da host del nostro report server. L'interfaccia IRemoteServer, l'oggetto Session e altre informazioni comuni tra client e server sono state inserite nel progetto CommonInterfaces.csproj; l'implementazione vera e propria del report server è definita nel progetto Server.csproj ed, infine, il client del nostro mini-sistema è definito nel progetto TestClient.csprorj. Per il corretto test e debugging dell'applicazione dall'IDE è dunque necessario avviare contemporaneamente sia l'host, che tiene in piedi il servizio, che il client. Tutti i progetti descritti sono contenuti nell'unica solution Remoting.sln; Visual Studio 2005 Express consente l'avvio e l'esecuzione contemporanea di più progetti presenti in una stessa solution, come mostrato in Figura 5. Una breve introduzione a .NET Remoting è presente nel Riquadro.

IL CLIENT CHE "CONSUMA" I REPORT

Il TestClient già introdotto nel paragro precendente è estremamente semplice: innanzitutto è un client di remoting che consuma il servizio IRemoteServer. La prima operazione necessaria

```
private void cmdReport_Click(object sender, EventArgs e)
{
    IRemoteServer srv = (IRemoteServer)Activator.GetObject(typeof(IRemote
                srv {System.Runtime.Remoting.Proxies.__TransparentProxy}
    IDictionary cfg = new Hashtable();
    ProcessReport repDel = new ProcessReport(srv.ProcessReport);
```

Fig. 6: Una tooltip a runtime sulla variabile srv che mostra come l'oggetto a cui punta è in realtà un server di remoting.

alla corretta interazione col server è l'autenticazione, atto con cui il client si fa riconoscere dal server e ne richiede l'avvio di una nuova sessione di funzionamento. Tale atto si consuma attraverso l'invocazione del metodo Login di IRemoteServer, come già descritto sopra:

```
private void cmdLogin_Click(object sender
                                    , EventArgs e)
{
    IRemoteServer srv = (IRemoteServer)Activator.
                    GetObject(typeof(IRemoteServer),
        "tcp://localhost:8737/remotingData/my
                                    FirstService");
    sessionId = srv.Login(txtUsername.Text,
                                txtPassword.Text);
    if (_sessionId != Guid.Empty)
    {
        cmdReport.Enabled = true;
        cmdRequestLock.Enabled = true;
        cmdRemoveLock.Enabled = true;
        cmdBreakableReport.Enabled = true;
    }
}
```

L'invocazione del server avviene attraverso il metodo GetObject dell'oggetto Activator del .NET Framework, operazione che server proprio a restituire un riferimento ad un oggetto remotamente invocabile ad un URI specifica (nell'esempio questo indirizzo è tcp://localhost:8737/remotingData/myFirstService). Da quel

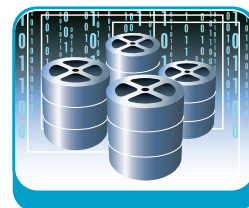
momento in poi l'oggetto IRemoteServer restituito dalla GetObject sarà utilizzabile normalmente come se fosse un oggetto istanziato direttamente e quindi presente nello stesso AppDomain dell'applicazione.

Ed eccoci finalmente giunti alla invocazione del report vero e proprio attraverso il ben noto metodo ProccessReport. Il client recupera l'istanza remota del server col solito GetObject, istanzia un Hashtable che sarà il dizionario di parametri da passare al report remoto, eventualmente definisce i parametri da passare ed è pronto ad invocare il report. Osserviamone il codice:

```
private void cmdReport_Click(object sender,
                                          EventArgs e)
{
   IRemoteServer srv = (IRemoteServer)Activator.
                  GetObject(typeof(IRemoteServer),
      "tcp://localhost:8737/remotingData/myFirst
                                         Service");
   IDictionary cfg = new Hashtable();
   cfg["parametro1"] = 13;
   cfg["parametro2"] = "Titolo per ioProgrammo";
   Stream stream = srv.ProcessReport(_sessionId,
                                             cfg);
   frmReportPreview preview = new
                    frmReportPreview(stream, cfg);
   preview.Show();
}
```

Ed ecco, infine, il codice del costruttore del form di preview del report, invocato nel metodo precedente:

```
private void CopyStream(Stream input, Stream ou
                                             put)
{
   input.Position = 0;
   byte[] bytes = new byte[4096];
   int i;
   while ((i = input.Read(bytes, 0, bytes.Length))
                                           != 0)
   {
      output.Write(bytes, 0, i);
   }
   output.Position = 0;
}
public frmReportPreview(Stream stream,
                      IDictionary parameters)
{
   InitializeComponent();
   MemoryStream newStream = new Memory
                                       Stream();
   CopyStream(stream, newStream);
   viewer1.Document.Load(newStream);
   viewer1.Show();
}
```

Lo stream proveniente dal server, contenente il report perfettamente processato, non può essere fruito direttamente ma deve essere prima copiato in uno stream gemello in locale, operazione che effettueremo con una semplice CopyStream che travasa il contenuto dello stream sorgente nella destinazione effettuando una copia byte per byte. Lo stream così ottenuto viene deserializzato con il metodo Load dell'oggetto Document di ActiveReports e il gioco è fatto.

MICROSOFT .NET REMOTING

.NET Remoting è la soluzione .NET al problema delle invocazioni remote e cioè all'invocazione tra processi distinti. Allo scopo di semplificare questo tipo di gestione remotizzata, .NET introduce il concetto di Application Domain: tipicamente un ambiente nativo tradizionale prevede il concetto di applicazione e di processo; quest'ultimo altro non è che una zona isolata nella memoria del sistema in cui viene caricata l'applicazione e in cui viene riservata un'area di memoria per il corretto funzionamento dell'applicazione stessa. I processi sono tipicamente fortemente isolati: non possono direttamente accedere alle risorse di un altro processo e sono protetti dai tentativi di accesso più o meno coscienti degli altri processi. D'altro canto si rende talvolta necessaria la comunicazione tra due proces-

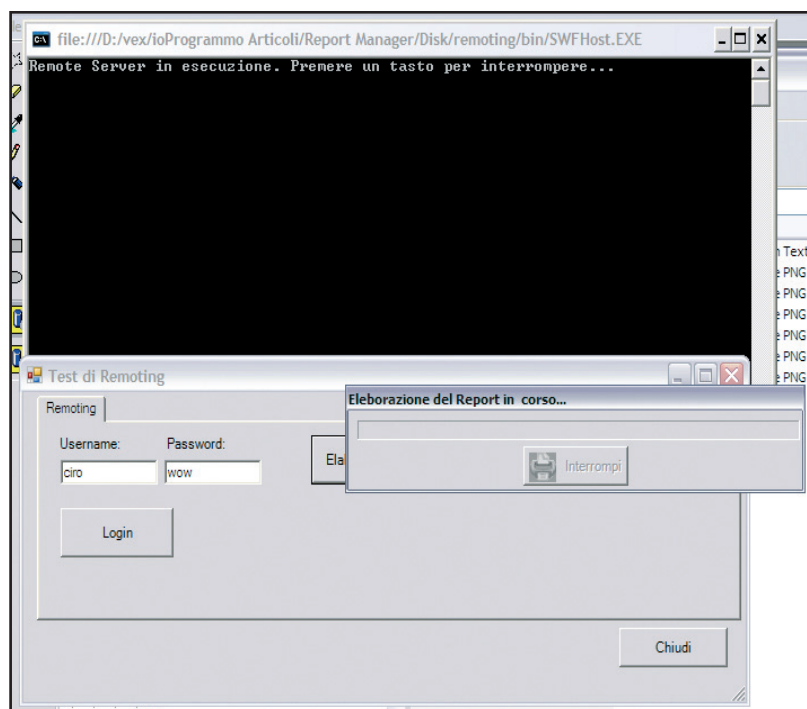



Fig. 7: *La spartana interfaccia del nostro client di test*

si e così il sistema operativo prevede i cosiddetti meccanismi di RPC (Remote Procedure Call). Essi in genere sono molto lenti e spesso complessi da gestire.
.NET consente di creare più Application Domain all'interno dello stesso processo fisico, massimizzandone così le prestazioni e riducendone così l'onere di gestione da parte del sistema operativo. Quando si prova ad invocare un oggetto presente nell'Application Domain (appdomain, per brevità) corrente, è sufficiente una tradizionale chiamata locale. Se invece l'oggetto è presente in un diverso appdomain si rende necessaria una chiamata remota. In tal caso sul client verrà creato un proxy, cioè un'istanza della classe TransparentProxy, mentre sul server viene creato uno stub. La potenza e la caratteristica innovativa di .NET Remoting, rispetto ad altre architetture di remotizzazione concorrenti, è che tale processo è completamente automatico e non è quindi richiesto allo sviluppatore l'onere di codificare tali strati proxy/stub.
Una comunicazione .NET Remoting prevede la presenza di un host e cioè di un'applicazione server che deve rispondere alle richieste, e di un client. La loro configurazione è molto semplice. Tutto, infatti, si basa sul file di configurazione

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.runtime.remoting>
    <customErrors mode="off" />
    <application name="remotingData">
        <lifetime
            leaseTime="100D"
            sponsorshipTimeOut="3M"
            renewOnCallTime="5M"
            leaseManagerPollTime="1M"
        />
        <channels>
            <channel ref="tcp" port="8737">
                <serverProviders>
                    <formatter ref="binary" typeFilterLevel="Full" />
                </serverProviders>
            </channel>
        </channels>
        <service>
            <wellknown mode="Singleton"
                type="ioProgrammo.RemotingServices.InterfaceServer, Server"
                objectUri="myFirstService" />
        </service>
    </application>
  </system.runtime.remoting>
</configuration>
```

Fig. 8: ***Il file di configurazione di remoting dell'host***

che, opzionalmente, può accompagnare le applicazioni .NET di tipo WinForms, Console o Windows Service, cioè il file App.config, oppure, se si usa IIS come host, il file Web.config. L'unico requisito è che l'applicazione, e quindi il processo Windows che ha generato, resti in vita per tutto il tempo. Il file può essere aggiunto nel progetto da Visual Studio .NET e, in fase di compilazione, viene copiato nella directory di compilazione del progetto, rinominandolo come l'eseguibile ma con l'estensione finale .config (es. miaApp.exe.config). Lo stesso discorso vale per i client. In questo file basta descrivere quale sia l'oggetto da pubblicare e il gioco è fatto, perché il client, semplicemente provando ad istanziare l'oggetto server pubblicato, grazie alle informazioni che rileva dal suo file di configurazione, effettua direttamente ed in modo trasparente la chiamata Remoting. In **Figura 8** e **Figura 9** sono mostrati rispettivamente i file di configurazione dell'host e del client.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.runtime.remoting>
    <customErrors mode="off" />
    <application name="remotingData">
        <lifetime
            leaseTime="100D"
            sponsorshipTimeOut="3M"
            renewOnCallTime="5M"
            leaseManagerPollTime="1M"
        />
        <channels>
            <channel ref="tcp" port="8737">
                <serverProviders>
                    <formatter ref="binary" typeFilterLevel="Full" />
                </serverProviders>
            </channel>
        </channels>
        <service>
            <wellknown mode="Singleton"
                type="ioProgrammo.RemotingServices.InterfaceServer, Server"
                objectUri="myFirstService" />
        </service>
    </application>
  </system.runtime.remoting>
</configuration>
```

Fig. 9: ***Il file di configurazione di remoting del client***

CONCLUSIONI

Abbiamo realizzato un completa soluzione a basso costo per produrre report lato server usando il prodotto gratuito Microsoft Visual Studio Express C# Edition il prodotto di reportistica ActiveReports .NET 3.0 dal costo di poche centinaia di euro, risparmiando così cifre consistenti necessarie all'acquisto di una soluzione di reportistica distribuita già preconfezionata. Nel prossimo numero evolveremo questa architettura per realizzare una più sofisticata interazione tra client e server anche durante la stessa elaborazione del report, ad esempio per interromperne l'esecuzione. Realizzeremo, cioè, un semplice sistema di sincronizzazione tra client e server. Intanto siete già pronti ad utilizzare questa nuova soluzione nelle vostre applicazioni.

Vito Vessia

Fig. 8: ***Il nostro report, elaborato sul server, in bella mostra sul client.***

L'AUTORE

Vito Vessia progetta e sviluppa applicazioni e framework in .NET, COM(+) e Delphi occupandosi degli aspetti architetturali. Scrive da anni per le principali riviste italiane di programmazione ed è autore del libro "Programmare il cellulare", Hoepli, 2002, sulla programmazione dei telefoni cellulari connessi al PC con protocollo standard AT+. Può essere contattato tramite e-mail all'indirizzo vvessia@ioprogrammo.it.

DATASTAGE: GESTIONE COMPLETA E GRAFICA

IBM DATASTAGE È IL PROGRAMMA CHE PERMETTE DI GESTIRE E MANIPOLARE I DATI SU QUALSIASI DATABASE SERVENDOSI SOLO DI UN'INTERFACCIA GRAFICA SEMPLICE DA USARE. RENDENDO COSÌ IL MITO DELLA TRASPARENZA UNA REALTÀ

In tutti i progetti la parte dei dati è sempre presente e assai delicata. Sono i dati ad essere visualizzati, modificati, inseriti. I dati determinano gli stipendi delle persone, i costi dei prodotti, le informazioni sanitarie di tutti. È da questa importanza che nascono strumenti in grado di gestire e trasformare i dati, anche provenienti da sistemi informatici diversi. *DataStage* è un programma che consente questo tipo di operazioni, note con il nome di *ETL*, *extraction*, *transformation*, *loading* (estrazione, trasformazione e caricamento). Il vantaggio principale di *DataStage* è di essere grafico e quindi, davvero tutti, possono usarlo da subito con grande facilità, eseguendo operazioni che richiederebbero, altrimenti, molte righe di codice. Tra l'altro, conoscerlo bene vuol dire avere una validissima carta da giocarsi nel mondo del lavoro. Questo articolo ha lo scopo di mostrare gli elementi caratteristici per dare la possibilità a tutti di usare *IBM DataStage* immediatamente, attraverso tre esempi che illustrano come usare tabelle, file e trasformazioni per gestire i dati in modo completo.

DATI SU TABELLE

Il flusso sul quale si fonda l'ETL, è l'estrazione dei dati da una sorgente, la loro trasformazione e il caricamento del risultato in una destinazione. Usando DataStage si può eseguire questa sequenza di operazioni in maniera molto semplice, usando pochi click di mouse. Generalmente, per memorizzare i dati, si usano tabelle e file. In questo esempio si vedrà come estrarre i dati da una tabella, eseguire una trasformazione elementare e inserire i risultati in un'altra tabella.

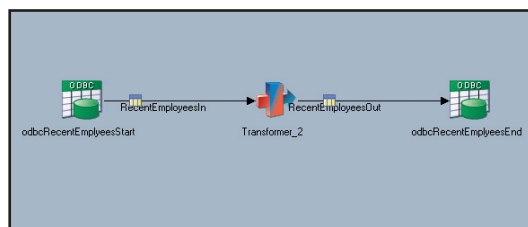



Fig. 1: Esempio odbcEmplyee.

REQUISITI

L'obiettivo da raggiungere è mostrato in fig. 1 ed evidenzia come l'accesso alle tabelle sia di tipo ODBC. E proprio la definizione del database è il primo passo da compiere. In allegato è presente un database Access che sarà la base di partenza per gli esempi.

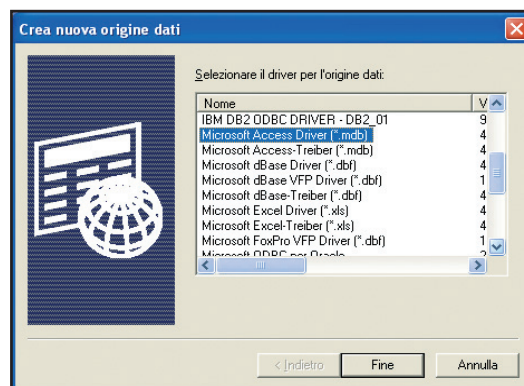



Fig. 2: Definizione ODBC sul computer.

Definendo la sorgente ODBC, anche le tabelle saranno accessibili da DataStage.

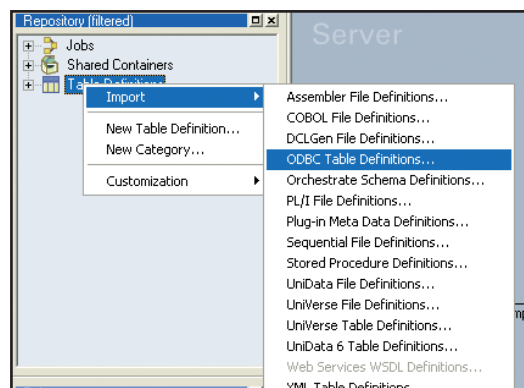



Fig. 3: Definizione ODBC in DataStage.

Conclusa la definizione delle tabelle, si passa alla creazione del Server Job vero e proprio. Si inseriscono i due elementi relativi ai collegamenti ODBC, uniti da una trasformazione. A questo punto, quello che si ottiene è la fig. 3, nella quale si devono impostare le informazioni specifi-

che dei vari elementi. Iniziando dalla prima tabella, cliccando due volte, si apre una finestra che permette di definire le caratteristiche generali del database e quelle specifiche della tabella. In particolare si sceglierà Employee come tabella dalla quale prendere i dati e si caricherà la definizione della tabella.

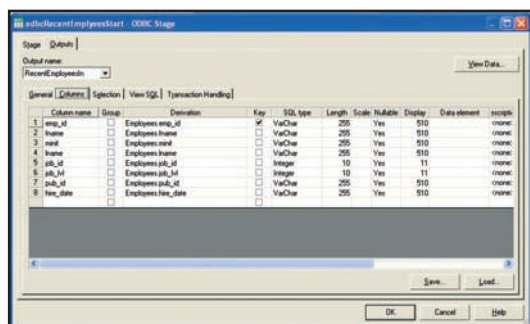



Fig. 4: **Colonne della tabella Employee.**

Per essere sicuri di aver collegato la tabella correttamente, si può spingere il tasto View Data.

odbcEmplyee..odbcRecentEmplyeesStart.RecentEmployeesIn - Data Browser

emp_id	fname	minit	lname	job_id	job_lvl	pub_id
PMA42628M	Paolo	M	Accorti	13	35	0877
PSA89086M	Piero	S	Astici	14	89	1389
VPA30890F	Vittoria	P	Ashworth	6	140	0877
L-B31947F	Liliana		Bellucci	7	120	0877

Fig. 5: **Visualizzazione dati della tabella.**

A questo punto il primo passo è concluso: la prima tabella è pronta. Come detto all'inizio, la trasformazione è molto semplice e opera un filtro sui dati scegliendo solo quelli con data maggiore di "01-01-90".

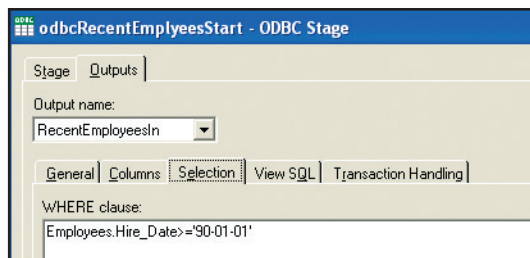



Fig. 6: **Selezione dei dati.**

Le informazioni sulle colonne sono propagate fino alla tabella finale.

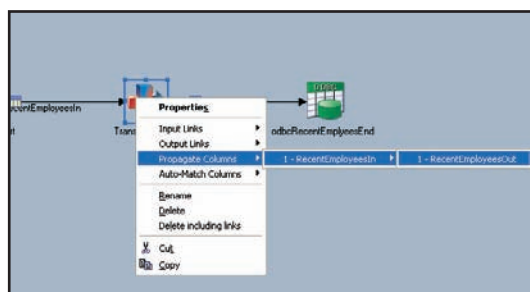



Fig. 7: **Propagazione metadati.**

In questo modo anche il secondo passo è concluso, avendo definito la trasformazione. Resta l'ultima tabella e, a questo punto, la procedura ricalca quanto fatto per la tabella precedente. A volte può capitare di voler prendere e trasformare i dati e inserirli in una nuova tabella che non esiste nel database: DataStage permette di farlo facilmente. Per dire che la tabella dovrà essere creata, basta impostare la condizione Create table in target database e spingere il tasto Create DDL, che contiene la definizione della tabella stessa, con nome e tipo dei campi.

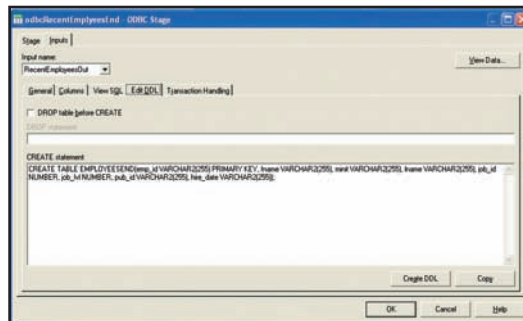



Fig. 8: **Istruzione per creare la tabella.**

A questo punto, anche il terzo e ultimo passo è completato e non resta che vedere all'opera quanto realizzato. Dopo aver salvato il tutto, si esegue la compilazione e poi l'esecuzione. Il risultato mostra gli obiettivi raggiunti, e cioè l'estrazione dei dati, la loro trasformazione e il caricamento nella tabella finale, specificando quante righe sono state inserite.
Questo esempio è un primo flusso ETL che sfrutta le tabelle e le loro definizioni ODBC. Questa modalità di accesso è la più generica e vale per qualsiasi database. Volendo ci sono anche gli accessi diretti ai vari Oracle, DB2, Informix, Sybase ecc. Già da adesso si può iniziare a usare DataStage per prendere, gestire dati e inserirli in tabella. E se non si volesse usare solo tabelle?

DATI SU FILE

Nell'esempio precedente, sia la sorgente che la destinazione dei dati erano tabelle. In altri scenari tipici, accade che i dati provengano da un altro sistema e siano scritti in file. Oppure, in modo analogo, che si debba prendere i dati da tabella, trasformarli, e avere i risultati formattati in file. In questo esempio il processo ETL estrarrà i dati da 2 file, li trasformerà e li caricherà su un altro file oppure su tabella. In particolare, la trasformazione selezionerà quali record inserire nel file e quali in tabella e, in quest'ultimo caso, eseguirà insert o update. Questo esempio utilizza più elementi e da la possibilità di sfruttare me-

NOTA

ETL

Il processo di estrazione dei dati, extraction, è il processo usato per ottenere dati da database, archivi, file. Poiché i dati possono venire da sorgenti diverse, la trasformazione, transformation, è il processo per rendere coerenti tutti i dati. Il processo finale di caricamento, load, serve per inserire i dati nel sistema informatico finale.

glio DataStage mantenendo sempre una notevole facilità di realizzazione.

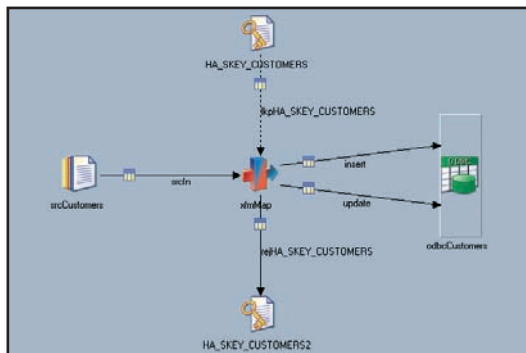



Fig. 9: **Esempio tranCustomers.**

Il primo elemento da impostare è srcCustomer, un file. I file sono configurabili in maniera molto simile alle tabelle. Con un doppio click sull'elemento si apre la finestra per personalizzarlo, specificando a quale file fisico presente sulla macchina corrisponde e, come per le colonne delle tabelle, quali sono i metadati. Anche in questo caso, come per le tabelle, si può visualizzare i dati contenuti nel file che è il metodo migliore per verificare che tutte le impostazioni siano corrette. L'altro file in ingresso alla trasformazione è HA_SKEY_CUSTOMERS, di tipo hash. I metadati corrispondenti completano la definizione del file.

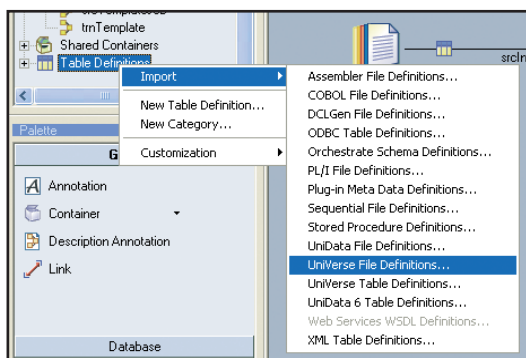



Fig. 10: **Definizione dei file.**

Il file HA_SKEY_CUSOTMERS2 ha impostazioni simili al file precedente. Fin qui si è visto che i file hanno configurazioni del tutto analoghe alle tabelle. Si può adesso iniziare ad usare una trasformazione. L'obiettivo è di verificare se esistono record uguali nei due file o meno. Nel primo caso, si esegue update sulla tabella; nel secondo caso, invece, si inserirà il record nel file HA_SKEY_CUSOTMERS2 e in tabella. Questa volta, come dimostra la fig. 9, le operazioni saranno di due tipi: di inserimento, se il record non è presente nella tabella, di aggiornamento, in caso contrario. Doppio click per aprire i parametri della trasformazione e selezionando Constraints si possono immettere le condizioni per capire quando effettuare insert e update in tabella e quando invece scartare il record inserendolo nel file. I valori dai inserire sono, per insert e rejHA_SKEY_CUSTOMERS, (IsNull(lkpHA_SKEY_CUSTOMERS.CustomerID)) mentre per update, Not(IsNull(lkpHA_SKEY_CUSTOMERS.CustomerID)).

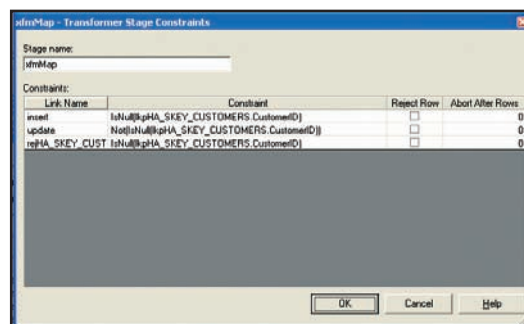



Fig. 11: **Vincolo della trasformazione.**

Il risultato finale è in fig. 11. Per costruire il vincolo si possono sfruttare anche le funzioni di DataStage alle quali si accede cliccando il campo del vincolo e poi il tasto sulla destra che fornisce tutte le opzioni per la definizione del vincolo stesso. A questo punto si sa cosa deve essere fatto dei record in ingresso e serve ancora dichiarare cosa andrà in uscita. Ritornando alle proprietà della trasformazione, come prima cosa notiamo i molti elementi in rosso che ci suggeriscono i campi da sistemare. Iniziando da CustomerID, semplicemente trascinando il valore da srcln nell'omonimo della tabella sottostante.

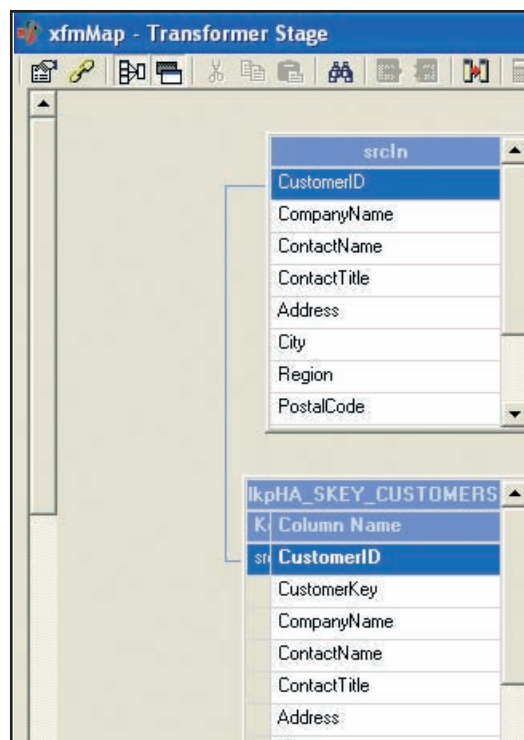



Fig. 12: **Collegamento tra campi.**

Come spesso accade in questi casi, vale più un'immagine, fig. 12, di mille parole. A questo punto, cliccando su insert e poi su Load Column Definition, importiamo i metadati della tabella Customers. Grazie al tasto Column Auto-Match, si ottiene la corrispondenza tra srcIn e insert. Eseguendo la stessa sequenza per update e rejHA_SKEY_CUSTOMERS, si ottengono gli stessi risultati.

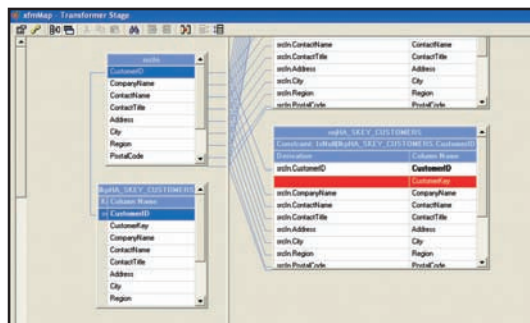



Fig. 13: ***Collegamenti tra campi.***

Dalla fig. 13, si può apprezzare ancora qualcosa in rosso da mettere a posto. Accanto al valore CustomerKey nei riquadri relativi a insert e rejHA_SKEY_CUSTOMERS, basta inserire @INROWNUM sia scrivendolo che aiutandosi con le funzioni di DataStage da scegliere tra quelle proposte cliccando l'apposito tasto. In questo modo si dichiara che il campo è un numero progressivo. In caso di update, invece, il valore CustomerKey viene direttamente dall'omonimo campo presente nel riquadro a sinistra, lkpHA_SKEY_CUSTOMERS ed il legame si ottiene trascinando il campo stesso da sinistra a destra. Il risultato è visibile in fig. 14. A questo punto la trasformazione è completa e manca solo la tabella di destinazione. Questa tabella è relativa a Customers ed esegue operazioni diverse: in caso di insert effettuerà Insert rows without clearing, in caso di update, invece, Update existing rows only. A questo punto è tutto completo, si deve solo compilare, eseguire e godersi il risultato. Con questo esempio si è entrati nei dettagli dei file e delle trasformazioni mostrando come il processo sia tutto grafico ed estremamente potente. Adesso si può impegnare di più Datastage.

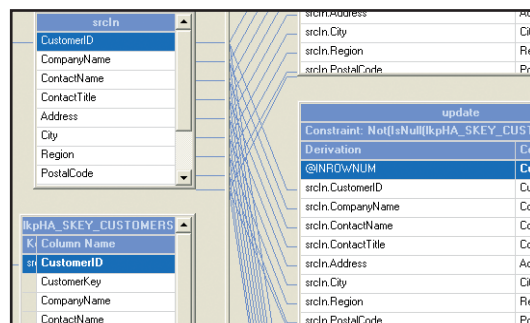



Fig. 14: ***Collegamenti per eseguire update.***

TRASFORMAZIONI AVANZATE

Ormai gli elementi basilari di DataStage sono noti e ci si può divertire ad approfondirne qualcuno in particolare. Nel prossimo esempio si entrerà maggiormente nei dettagli delle trasformazioni e delle loro enormi possibilità e di come siano di facile utilizzo. Mai capitato di dover gestire date in diversi formati? In America mettono prima l'anno (1996 11 22); qualcuno usa il simbolo "/" per separare i campi (22/12/96): insomma la stessa data assume forme assai diverse. E può anche essere utile calcolare le differenze tra date. L'esempio successivo risolve questi problemi e anche qualcun altro.

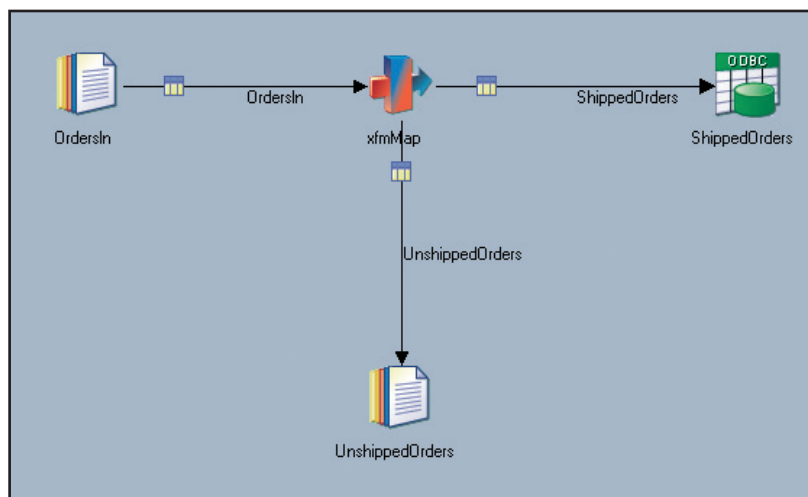



Fig. 15: ***Esempio tranOrder.***

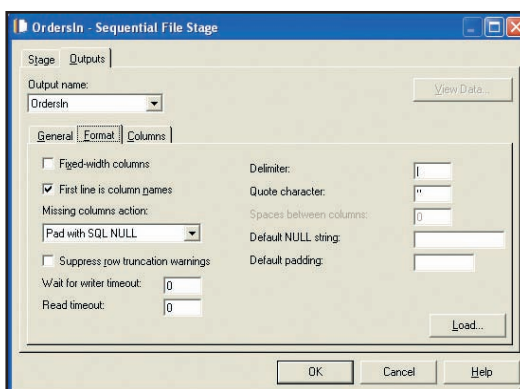



Fig. 16: ***Proprietà del file***

Come si può vedere in fig. 15, il flusso ETL è molto semplice per potersi concentrare sulla trasformazione. Come detto negli esempi precedenti, l'elemento OrdersIn deve essere collegato al file fisico corrispondente e deve essere associato ai metadati della struttura Orders. Questa volta il simbolo che delimita i campi è "|".
Per dichiarare che i campi del file relativo a UnshippedOrders sono gli stessi del file OrdersIn,

DATA WAREHOUSE

I database servono per gestire dati che cambiano molto velocemente e sono acceduti e modificati in maniera concorrente. Il data warehouse estrae dati da uno o più sorgenti, anche database, per contenere dati stabili, in grande quantità, pronti per essere analizzati.

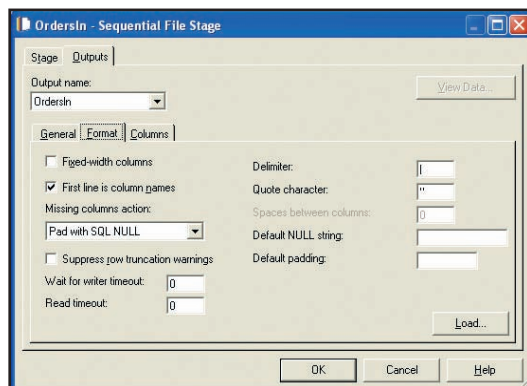
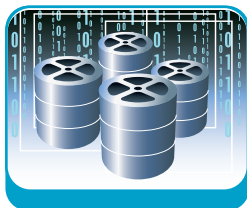



Fig. 17: ***Propagazione dei metadati.***

serve cliccare con il tasto destro e propagare i metadati. Invece per la tabella ShippedOrders si possono definire i campi manualmente andando a riempire i valori di Columns, fino ad ottenere la situazione di fig. 17.

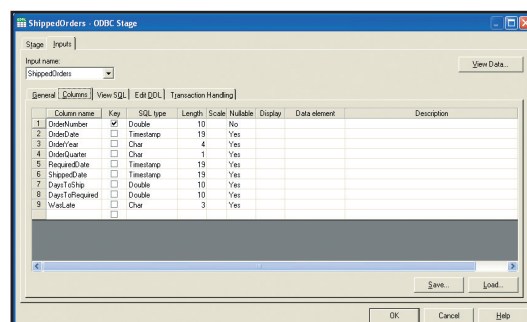



Fig. 18: ***Colonne della tabella Orders.***

Una volta finito, si deve spingere il tasto Create DDL per rendere effettive le modifiche. Sistemati gli elementi file e tabella, si può passare alla trasformazione. Come nell'esempio precedente, si imposta la condizione in base alla quale i dati andranno nella tabella piuttosto che nel file. Come suggeriscono i nomi del file e della tabella di fig. 15, il fattore discriminante sarà se l'ordine ha la data ShippedDate valorizzata o meno.

```
10550|GODOT|MGK44605M|1997 05 28|97/06/25  |19970606  |   3
10882|SAVEL|POK93028M|1998 02 11|98/03/11  |19980220  |   3
10473|ITRAD|PMA42628M|1997 03 13|97/03/27  |19970321  |   3
10705|HILAR|MAS70474F|1997 10 15|97/11/12  |19971118  |   2
11061|GLAFO|SKO22412M|1998 04 30|98/06/11  ||  3|    14.01
10558|AROHO|M-P91209M|1997 06 04|97/07/02  |19970610  |   2
10975|BODOM|DWR65030M|1998 03 25|98/04/22  |19980327  |   3
10546|VICTY|POK93028M|1997 05 23|97/06/20  |19970527  |   3
10717|FRANK|MMS49649F|1997 10 24|97/11/21  |19971029  |   2
```

Fig. 19: ***Righe del file Orders.src.***

Come si può vedere dalla fig. 19, la riga blu ha un campo valorizzato in più rispetto a quella gialla. In particolare, dove per la riga blu c'è il valore "19971118" (ancora un altro formato di data…), nel corrispondete campo della riga gialla non è presente nessun valore ("||"). L'obiettivo, quindi, è di mettere la riga blu, e tutte quelle con il campo valorizzato, in tabella, e le altre nel file. Per specificare questo comportamento alla trasformazione, si deve premere il tasto Constraint e impostare la condizione di campo vuoto, null.

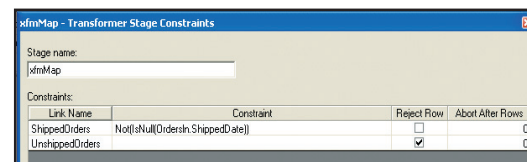



Fig. 20: ***Vincolo sulla data.***

Molto più facile a vedersi. In questo modo si impone che il record finirà nella tabella ShippedOrders se il campo ShippedDate è presente nel file di ingresso, OrdersIn, è non null, cioè non vuoto, ossia, eliminando la doppia negazione, se il campo ShippedDate è valorizzato. In caso contrario il record verrà spostato nel file UnshippedOrders. A questo punto si possono modificare le date e formattarle opportunamente per ottenere i dati finali, su file e tabella, tutti coerenti. In questa fase subentra un concetto noto della programmazione: se si vuole passare da un certo valore ad un altro attraverso una serie operazioni, si possono usare le variabili. E poiché l'obiettivo dell'esempio è proprio di prendere una data da file, modificarla, eseguire dei calcoli e poi mettere i risultati in uscita, allora le variabili sono l'ideale. In questo caso, infatti, si calcolerà se l'ordine può essere soddisfatto, verificando che la data RequiredDate sia maggiore di ShippedDate. Il problema, in questo caso, è di prendere le due date in ingresso che sono in formati non compatibili, trasformarle in uno stesso formato, confrontarle, e mettere il risultato (che è un valore Yes/No) in tabella. Quindi, il processo di trasformazione si compone di 2 fasi. La prima di definizione delle variabili e la seconda delle trasformazioni vere e proprie che fanno uso delle variabili. Per dichiarare le variabili c'è un apposito riquadro nel quale indicare il nome e la conversione da effettuare. Ad esempio, guardando la riga blu nella fig. 19 e le variabili usate di fig. 21, la prima data, "1997 10 15", si riferisce a OrdersIn.OrderDate e il suo formato ha come delimitatore uno spazio con 4 campi per anno, 2 per mese e 2 per giorno. In DataStage il formato relativo è "DYMD[4,2,2]" che tiene conto proprio del carattere che delimita (dopo la D di Delimiter c'è uno spazio), dell'ordine e di quanti valori è composto ogni singolo campo. Analoghi sono i riferimenti a "97/11/12" con "D/YMD[2,2,2]" in cui il simbolo che delimita è "/" e che per Year, anno, si usano 2 caratteri, cosi

Fig. 21: ***Dichiarazione variabili.***

come per Month, mese, e Day, giorno.
Una volta dichiarate le variabili, non resta che

ShippedOrders
Constraint: Not(IsNull(OrdersIn.ShippedDate))

Derivation	Column Name
OrdersIn.OrderID	OrderNumber
Oconv(OrderDt,"D-YMD[4,2,2]")	OrderDate
YEAR.TAG(OrderDt)	OrderYear
Oconv(OrderDt,"DQ")	OrderQuarter
Oconv(ReqDt,"D-YMD[4,2,2]")	RequiredDate
Oconv(ShipDt,"D-YMD[4,2,2]")	ShippedDate
ShipDt - OrderDt	DaysToShip
ReqDt - ShipDt	DaysToRequired
IF ReqDt > ShipDt then 'no' Else 'Yes'	WasLate

Fig. 22: ***Definizione delle operazioni sulle colonne.***

usarle.
Come si può notare dalla fig. 22, c'è l'utilizzo delle variabili create in precedenza: OrderDt, ReqDt e ShipDt. Per avere nella tabella di destinazione le date coerenti, si è eseguita una ulteriore conversione, del tutto simile a quella usata nella definizione delle variabili, per impostare il formato come in "1997-10-15". Da osservare anche che DaysToRequired, usando le variabili, è una normale differenza. E che WasLate è il risultato di una espressione if, then, else e ritorna un valore Yes/No.
Una volta impostato sorgenti, destinazioni e trasformazione è il momento di un po' di soddisfazione. Dopo aver compilato il tutto, si esegue il flusso. Il risultato dovrebbe essere quello

Fig. 23: ***Risultato finale***

di fig. 23.
Già dalla figura si può vedere come la maggior parte delle transazioni abbiano il campo ShippedDate valorizzato e vadano quindi in tabella. Andando ad indagare nei valori inseriti, si possono apprezzare tutte le operazioni eseguite. Effettivamente il campo ShippedDate è sempre valorizzato (altrimenti il record sarebbe finito nel file) e tutte le date hanno lo stesso formato. Il campo DaysToRequired è esattamente la differenza tra ShippedDate e RequiredDate, e il campo WasLate riferisce se la data RequiredDate è minore di ShippedDate.
Con questo esempio si può apprezzare quanto sia facile effettuare operazioni che, altrimenti, richiederebbero molte righe di codice e con le quali è sempre facile cadere in errore.

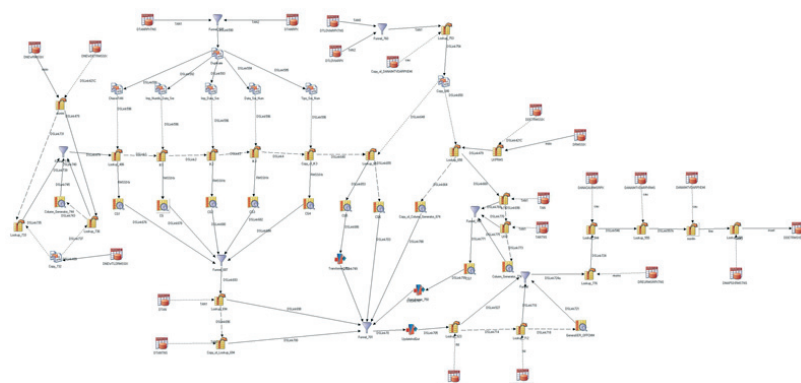
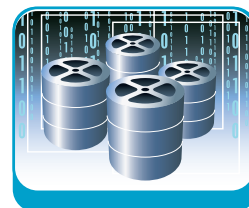



Fig. 24: ***Esempio da esperti.***

CONCLUSIONI

Attraverso i tre esempi si è potuto verificare la potenza di un programma che fa della semplicità e dello sviluppo grafico i suoi punti di forza. Chiunque, anche chi non conosce nulla di programmazione, può usarlo da subito ottenendo, in brevissimo tempo, risultati eccezionali.
Per concludere coerentemente con tutto l'articolo, l'immagine finale da l'idea di come gli esperti DataStage possano realizzare processi estremamente complessi, sfruttando sempre gli elementi base del prodotto.

Cristiano Bellucci

tranOrder..ShippedOrders.ShippedOrders - Data Browser

OrderDate	OrderYear	OrderQuarter	RequiredDate	ShippedDate	DaysToShip	DaysToRequired	WasLate
1997-03-04 00:00:00	1997	1	1997-04-01 00:00:00	1997-03-06 00:00:00	2	26	no
1998-03-20 00:00:00	1998	1	1998-04-17 00:00:00	1998-03-30 00:00:00	10	18	no
1997-05-26 00:00:00	1997	2	1997-06-23 00:00:00	1997-06-02 00:00:00	7	21	no
1997-08-13 00:00:00	1997	3	1997-09-10 00:00:00	1997-08-19 00:00:00	6	22	no
1997-10-27 00:00:00	1997	4	1997-11-24 00:00:00	1997-11-05 00:00:00	9	19	no
1997-03-14 00:00:00	1997	1	1997-04-11 00:00:00	1997-04-04 00:00:00	21	7	no
1998-03-26 00:00:00	1998	1	1998-04-23 00:00:00	1998-04-10 00:00:00	15	13	no
1998-04-30 00:00:00	1998	2	1998-05-28 00:00:00	1998-05-06 00:00:00	6	22	no

Fig. 24: ***Dati inseriti.***

NOTA

COMPONENTI DATASTAGE

Componenti client di DataStage sono Designer, lo strumento considerato nell'articolo che serve per realizzare i flussi ETL. Il Director serve per validare, schedulare, e controllare i processi in esecuzione. Administrator serve per gestire i progetti e gli utenti.

mato ha come delimitatore uno spazio con 4 campi per anno, 2 per mese e 2 per giorno. In Data-

ShippedOrders	
Constraint: Not(IsNull(OrdersIn.ShippedDate))	
Derivation	Column Name
OrdersIn.OrderID	OrderNumber
Oconv(OrderDt,"D-YMD[4,2,2]")	OrderDate
YEAR.TAG(OrderDt)	OrderYear
Oconv(OrderDt,"DQ")	OrderQuarter
Oconv(ReqDt,"D-YMD[4,2,2]")	RequiredDate
Oconv(ShipDt,"D-YMD[4,2,2]")	ShippedDate
ShipDt - OrderDt	DaysToShip
ReqDt - ShipDt	DaysToRequired
IF ReqDt > ShipDt then 'no' Else 'Yes'	WasLate

Fig. 22: ***Definizione delle operazioni sulle colonne.***

Stage il formato relativo è "D YMD[4,2,2]" che tiene conto proprio del carattere che delimita (dopo la D di Delimiter c'è uno spazio), dell'ordine e di quanti valori è composto ogni singolo campo. Analoghi sono i riferimenti a "97/11/12" con "D/YMD[2,2,2]" in cui il simbolo che delimita è "/" e che per Year, anno, si usano 2 caratteri, così come per Month, mese, e Day, giorno. Una volta dichiarate le variabili, non resta che usarle. Come si può notare dalla fig. 22, c'è l'utilizzo delle variabili create in precedenza: OrderDt, ReqDt e ShipDt. Per avere nella tabella di destinazione le date coerenti, si è eseguita una ulteriore conversione, del tutto simile a quella usata nella definizione delle variabili, per impostare il formato come in "1997-10-15". Da osservare anche che DaysToRequired, usando le variabili, è una normale differenza. E che WasLate è il risultato di una espressione if, then, else e ritorna un valore Yes/No. Una volta impostato sorgenti, destinazioni e trasformazione è il momento di un po' di soddisfazione. Dopo aver compilato il tutto, si esegue il flusso. Il risultato dovrebbe essere quello di fig. 23. Già dalla figura si può vedere come la maggior parte delle transazioni abbiano il campo ShippedDate valorizzato e vadano quindi in tabella. Andando ad indagare nei valori inseriti, si possono apprezzare tutte le operazioni eseguite.

Effettivamente il campo ShippedDate è sempre valorizzato (altrimenti il record sarebbe finito nel file) e tutte le date hanno lo stesso formato. Il campo DaysToRequired è esattamente la dif-

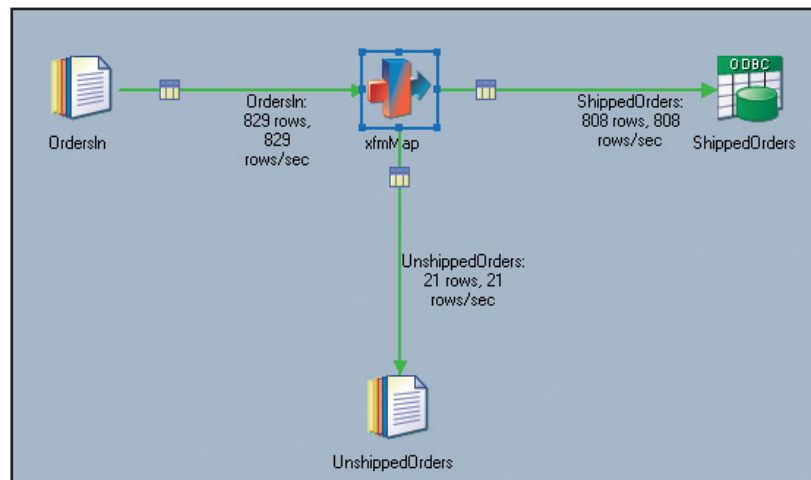



Fig. 23: ***Risultato finale***

ferenza tra ShippedDate e RequiredDate, e il campo WasLate riferisce se la data RequiredDate è minore di ShippedDate. Con questo esempio si può apprezzare quanto sia facile effettuare operazioni che, altrimenti, richiederebbero molte righe di codice e con le quali è sempre facile cadere in errore.

CONCLUSIONI

Attraverso i tre esempi si è potuto verificare la potenza di un programma che fa della semplicità e dello sviluppo grafico i suoi punti di forza. Chiunque, anche chi non conosce nulla di programmazione, può usarlo da subito ottenendo, in brevissimo tempo, risultati eccezionali. Per concludere coerentemente con tutto l'articolo, l'immagine finale da l'idea di come gli esperti DataStage possano realizzare processi estremamente complessi, sfruttando sempre gli elementi base del prodotto.

Cristiano Bellucci

COMPONENTI DATASTAGE

Componenti client di DataStage sono Designer, lo strumento considerato nell'articolo che serve per realizzare i flussi ETL. Il Director serve per validare, schedulare, e controllare i processi in esecuzione. Administrator serve per gestire i progetti e gli utenti.

tranOrder..ShippedOrders.ShippedOrders - Data Browser

OrderDate	OrderYear	OrderQuarter	RequiredDate	ShippedDate	DaysToShip	DaysToRequired	WasLate
1997-03-04 00:00:00	1997	1	1997-04-01 00:00:00	1997-03-06 00:00:00	2	26	no
1998-03-20 00:00:00	1998	1	1998-04-17 00:00:00	1998-03-30 00:00:00	10	18	no
1997-05-26 00:00:00	1997	2	1997-06-23 00:00:00	1997-06-02 00:00:00	7	21	no
1997-08-13 00:00:00	1997	3	1997-09-10 00:00:00	1997-08-19 00:00:00	6	22	no
1997-10-27 00:00:00	1997	4	1997-11-24 00:00:00	1997-11-05 00:00:00	9	19	no
1997-03-14 00:00:00	1997	1	1997-04-11 00:00:00	1997-04-04 00:00:00	21	7	no
1998-03-26 00:00:00	1998	1	1998-04-23 00:00:00	1998-04-10 00:00:00	15	13	no
1998-04-30 00:00:00	1998	2	1998-05-28 00:00:00	1998-05-06 00:00:00	6	22	no

Fig. 24: ***Dati inseriti.***

UN CLIENT FTP VIA WEB...WORK!

WEBWORK, IL FRAMEWORK CHE SI APPRESTA A SOPPIANTARE STRUTS, PERMETTE DI CREARE IN MANIERA SEMPLICE ED ELEGANTI APPLICAZIONI WEB SECONDO IL PATTERN MVC. IN QUESTO ARTICOLO ILLUSTREREMO UN ESEMPIO COMPLETO

Chiunque programmi applicazioni Web sa che è imperativo, pena il caos, adottare regole architetturali che semplifichino la separazione dei ruoli delle diverse componenti coinvolte. Queste regole sono sintetizzate dal pattern MVC, ovvero Model View Controller. Tale pattern prevede di suddividere (logicamente) la propria applicazione in tre blocchi: una parte gestisce i dati veri e propri (parte di elaborazione e trasformazione) ed è chiamata Model; un'altra si occupa della visualizzazione dei risultati (View) e la terza ed ultima parte si occupa di applicare le opportune trasformazioni/elaborazioni (ovvero sceglie quali componenti del Model applicare) e in che modo visualizzare i dati (scegliendo l'opportuna View): questa parte è chiamata Controller. In un ambiente J2EE la parte del Controller è, solitamente, svolta da una Servlet, quella delle View da pagine JSP e la parte Model da un insieme di JavaBean o EJB (**Figura 1**).

Fig. 1: *Il modello MVC in ambiente J2EE*

Visto l'ampio utilizzo del pattern sono nati numerosi framework che semplificato la scrittura di applicazioni secondo i suoi dettami. In particolare gran parte di essi realizza solo il Controller (in maniera generalizzata) e ne permette la configurazione e personalizzazione grazie a file esterni (di solito file XML). Inoltre ogni framework adotta proprie regole per la scrittura dei Model e il modo in cui essi comunicano i dati alle View. Non solo: i diversi framework offrono componenti di più alto livello di astrazione per semplificare la scrittura delle JSP (in pratica consentono di usare dei componenti e di non preoccuparsi della generazione del corrispondente codice HTML, generazione fatta al run-time dal framework stesso) e le modalità di comunicazione dei dati ai bean. Questa standardizzazione porta sia ad una semplificazione concettuale sia al riuso di componenti Model o View tra applicazioni diverse. Ma proprio la quantità di possibili framework può portare ad un disorientamento su quale scelta può essere più appropriata.

WEBWORK? STRUTS DI DOMANI!

Al di là di valutazioni soggettive, ciascuno può trovare particolarmente conveniente un framework piuttosto che un altro. WebWork (http://www.opensymphony.com/webwork) si sta affermando come uno dei framework di maggior interesse. In passato questo privilegio lo ha avuto Struts (http://struts.apache.org/), il famework MVC con la più ampia e solida base di utenti in ambiente J2EE. Ma ultimamente c'è stata una singolare convergenza tra i due progetti: Struts ha deciso che la prossima release del prodotto (Struts 2) manterrà il nome del progetto originario e, si spera, la stessa base di sviluppatori e la stessa community, ma si fonderà con WebWork e di quest'ultimo manterrà gran parte dell'architettura e della filosofia di funzionamento. Questo a dimostrazione delle ottime qualità di design di WebWork. Struts 2 per ora non è sufficientemente stabile e se ne sconsiglia l'uso in ambiente di produzione. Sul sito del progetto (vedi http://cwiki.apache.org/WW/home.html) si consiglia, per progetti che richiedono stabilità e, allo stesso tempo, pronti per una migrazione quanto più indolore possibile a Struts 2, di usare l'ultima release di WebWork: la 2.2.4 (al momento di scrivere l'articolo non sono previste ulteriori release;

anche l'ultima è solo una versione di correzione di bug e non aggiunge funzionalità rispetto alla 2.2.3).

STRUTTURA DI UN'APPLICAZIONE

Come tutte le webapp, l'applicazione che utilizza WebWork deve avere un descrittore chiamato web.xml e posizionato nella cartella WEB-INF del progetto. Accanto alle usuali dichiarazioni di versione, eventuali parametri di contesto e quant'altro, è necessario dichiarare un filtro:

```
<filter>
  <filter-name>webwork</filter-name>
  <filter-class>
com.opensymphony.webwork.dispatcher
                                    .FilterDispatcher
  </filter-class>
</filter>
```

questo filtro permette di far processare le richieste al framework WebWork: è il componente che realizza il controller per l'applicazione. Per farlo entrare in azione è necessario indicare a quali url esso risponde:

```
<filter-mapping>
  <filter-name>webwork</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

In questo caso viene invocato su tutte le url. Però solo quelle con suffisso .action faranno sì che esso proceda all'elaborazione della richiesta, a meno di settare il framework diversamente, indicando un pattern diverso. Questi e altri settaggi sono, a loro volta, presenti in un file XML di configurazione: xwork.xml. Siccome esso deve trovarsi nel classpath della webapp, il posto più appropriato ove inserirlo è in /WEB-INF/classes/. xwork.xml contiene tutti gli oggetti definiti dalla webapp e che devono essere gestiti da WebWork.

AZIONI

Ogni nuova richiesta di pagina (sia esso il submit di una form o un link verso un file della webapp), chiama in causa il controller di WebWork il quale deve determinare se per quella richiesta è di sua competenza. Se lo è, per rispondere, deve avere registrata un'azione.

Un'azione è una classe Java che implementa l'interfaccia com.opensymphony.xwork.Action, il cui unico metodo da dover implementare è *execute()*. Tale metodo verrà invocato dal framework. Spesso, anziché implementare tale interfaccia, si è soliti estendere la classe com.opensymphony.xwork. ActionSupport: così facendo, si eredita un insieme di metodi e attributi di grande utilità nello sviluppo, quali gestione degli errori, dei messaggi e così via.

INTERCEPTORS

Accanto alle azioni specifiche per determinate richieste, WebWork permette di invocare altri oggetti: gli interceptors. Essi sono classi stateless (ovvero non mantengono uno stato tra invocazioni successive) e possono essere invocati prima e dopo un'azione o un insieme qualuque di azioni (chi conosce l'architettura J2EE, riconoscerà che gli interceptors svolgono lo stesso compito dei filtri, infatti entrambi implementano il pattern “interceptor”). Usando questi interceptors è possibile, per esempio, validare l'input, intercettare eccezioni e così via; il tutto separando la gestione di questi aspetti rispetto alle azioni (e quindi applicando interceptor simili per azioni diverse e riutilizzandoli anche in altri progetti). Il framework dispone di numerosi interceptor predefiniti (alla pagina http://wiki.opensymphony.com/display/WW/Interceptors una loro lista) ciascuno dei quali può essere inserito in uno stack (dove per stack si intende un insieme di interceptors) e ciascuno stack essere associato a diverse url. L'ordine di esecuzione dei diversi

DOWNLOAD E INSTALLAZIONE

Supponendo di avere già un JDK installato (se ne consiglia la versione 1.4 o successiva) si può procedere con il download (pagina http://www.opensymphony.com/webwork/download.action) del file compresso (per l'uso si può prendere la versione compilata e non quella con i sorgenti): una volta scompattato il file ci si trova una struttura come quella mostrata in Figura 2. Essa contiene tutti i file necessari per realizzare le proprie applicazioni.

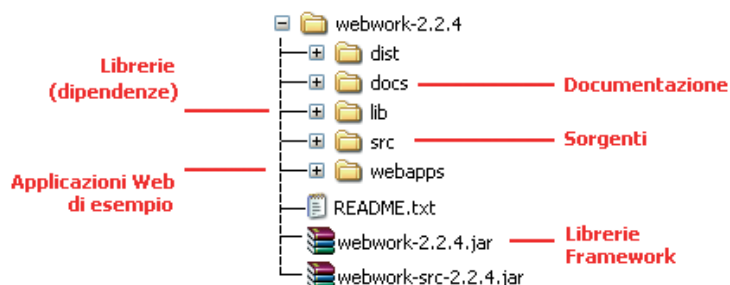



***Fig. 2:* Struttura di WebWork (in formato compilato).**

interceptors è dato dal loro ordine di dichiarazione nello stack (chi è dicharato per primo, viene anche eseguito per primo). Ogni stack è definito per nome (quello di default è chiamato "defaultStack").

COME "ASSEMBLARE" AZIONI E INTERCEPTORS

Definiti quelli che sono i mattoncini base del framework, resta da capire come essi possano essere assemblati tra loro all'interno del file /WEB-INF/classes/xwork.xml. Gran parte degli oggetti predefiniti (interceptors e stack standard) sono sempre gli stessi per la maggior parte delle applicazioni. Per questo è meglio lasciare i valori di default esterni al file e includerli con i seguenti tag iniziali:

```
<include file="webwork-default.xml"/>
<package name="default" extends="webwork-
                                  default">
<default-interceptor-ref name="defaultStack"/>
```

Essi, nell'ordine, indicano il file dove sono presenti le altre dichiarazioni, definiscono il package di default e qual è lo stack predefinito. Per curiosare sul file webwork-default.xml è necessario aprire il jar principale del framework (webwork-2.2.4.jar). Al suo interno si possono trovare tutti gli interceptor standard e i diversi stack predefiniti.

GRAFO DI NAVIGAZIONE

Il file xwork.xml conterrà le dichiarazioni di tutte le azioni. Esse, per forza di cose, sono specifiche all'applicazione in esame. Ogni dichiarazione di azione ha la seguente struttura:

```
<action name="nomeUnivocoAzione"
   class="nome.del.package.nomeClasse">
   <result name="success"
           type="dispatcher">nomeVista.jsp</result>
</action>
```

Questa dichiarazione lega insieme, in un nome univoco per l'azione, la classe che calcola i risultati (Model) con la vista che li visualizza (View). Si noti che il risultato ha, a sua volta, un nome. Possono esistere più risultati: in pratica, il Model setta qual è il risultato (per esempio può indicare se è andato tutto bene, se c'è stato un errore, se necessita di ulteriori informazioni e così via) e il Controller deciderà, grazie alla dichiarazione, qual è la vista appropriata da presentare. In particolare si realizza un grafo di navigazione grazie all'insieme di tutte le azioni dichiarate nel file.

LE VIEW

Le pagine che mostrano i risultati possono essere pagine JSP che fanno uso di tag personalizzati (libreria di tag) e di un linguaggio per esprimere espressioni di accesso ai dati chiamato Object Graph Navigation Language (OGNL).
Finora si è visto il significato dei diversi oggetti e come metterli in relazione tra loro. Ora è venuto il momento di scrivere del codice che realizza un esempio concreto.

UN ESEMPIO: PAGINA DI LOGIN

Praticamente ogni applicazione necessita dell'autenticazione dell'utente almeno per alcune delle sue pagine. Ecco come realizzare un accesso, con controllo dell'utente, usando WebWork e i suoi oggetti. Innanzitutto va dichiarato nel file WEB-INF/classes/xwork.xml il percorso di login; un esempio potrebbe essere:

```
<action name="login"
class="it.ioprogrammo.ftpwebwork.actions.welcom
                                      eAction">
 <result type="dispatcher">
  login.jsp
 </result>
</action>
```

Questo percorso sta a significare che l'azione login viene processata dalla classe welcome Action (potrebbe non far nulla o inserire dei messaggi di benvenuto) e il risultato è mostrato dalla pagina login.jsp. Serve almeno una seconda azione che mostra il risultato della login. Essa può essere:

CONTAINER IOC

Dietro le quinte di WebWork ci deve essere un conteiner per realizzare il pattern IoC (Inversion of Control). Di default WebWork ne fornisce uno ma è consigliato usare Spring (http://www.springframework.org/). In realtà l'uso di Spring passa praticamente inosservato. È sufficiente assicurarsi che la cartella WEB-INF/lib/ ne contenga le librerie (esse possono essere copiate dalla cartella /lib/spring/ dell'installazione di WebWork) e settare opportunamente il file /WEB-INF/applicationContext.xml che contiene le dichiarazioni dei bean gestiti dal container.

```
<action name="trylogin"
class="it.ioprogrammo.ftpwebwork.actions.
                                        loginAction">
 <result name="error" type="dispatcher">
  login.jsp
 </result>
 <result name="success" type="dispatcher">
  lista.jsp
 </result>
</action>
```

Al contrario di prima, il risultato finale, ovvero la pagina mostrata, dipende dal risultato della login: se è andato a buon fine (success) mostrerà la pagina lista.jsp, altrimenti ripresenterà la pagina di login. La login può, semplicemente, mostrare una form html con gli usuali campi di inserimento username/password:

```
<form action="trylogin.action" method="post">
 <input type="text" name=" nome" />
 <input type="password" name=" pwd" />
<input type="submit" value="Accedi" />
</form>
```

Si noti come il nome della action è lo stesso di quello indicato nel file xwork.xml, ma con estensione .action. Ora non resta che scrivere l'azione loginAction, che legge i parametri passati ed esegue (o simula) la login. La si potrebbe realizzare implementando il metodo execute e i set sui parametri che si vuole leggere dalla request (nome e pwd):

```
public class loginAction extends ActionSupport{
 private String nome;
 private String pwd;

 public void setPwd(String nuovo){
  pwd = nuovo;
 }

 public void setNome(String nuovo){
  nome = nuovo;
 }
public String execute() {
  if ("ivan".equals(nome()) &&
    "venuti".equals(pwd) )
   return SUCCESS;
  else
   return ERROR;
 }
}
```

Ci si potrebbe chiedere come avviene l'invocazione dei metodi “set” a partire dalla JSP; presto detto: non serve fare altro in quanto un interceptor legge ciascun parametro passato e cerca un omonimo metodo set. Quando in request c'è un parametro come “nome”, viene cercato (e se presente invocato) il metodo set“Nome”(String). L'interceptor in realtà fa molto di più: se trova un parametro del tipo: “nome1.nome2.nome3”, tenta di reperire un bean “nome1”, con al suo interno il bean “nome2” e, infine il metodo “setNome3”. Vediamo come sfruttare questa caratteristica per “isolare” le informazioni della login.

UN BEAN PER I PARAMETRI DI LOGIN

Accedendo via FTP si ha la necessità di almeno tre parametri: lo username/password di autenticazione e il nome del server FTP. Ecco che si potrebbe realizzare un bean che incapsula queste informazioni (non sono riportati i set/get per pwd e server; la classe completa è presente sul CD-Rom allegato insieme alla webapp di esempio):

```
public class login implements java.io.Serializable{
 private String nome;
 private String pwd;
 private String server;

 public login(){}

 public login(String nome, String pwd){
  this.nome = nome;
  this.pwd = pwd;
 }

 public String getNome() {
  return nome;
 }

 public void setNome(String nome) {
  this.nome = nome;
 }
// Altri setter/getter
}
```

Ora si può riscrivere la form di login come:

```
<form action="trylogin.action" method="post">
 <input type="text" name="myLogin.nome" />
 <input type="password" name="myLogin.pwd" />
 <input type="text" name="myLogin.server" />
 <input type="submit" value="Accedi" />
</form>
```

E modificare la classe loginAction in questo modo:

```
public class loginAction extends ActionSupport{
 private login myLogin;
 public login getMyLogin() {
  return myLogin;
```

```
 }
 public void setMyLogin(login myLogin) {
  this.myLogin = myLogin;
 }
//...
}
```

La logica è la stessa di prima, ma le diverse informazioni risultano incapsulate in un'unica classe e, per questo, molto più vicine al mondo Object Oriented rispetto alla soluzione precedente.

CLIENT FTP

Non resta che implementare il metodo execute() con l'effettiva connessione al server remoto. Per gestione di una comunicazione FTP (accesso, lista file, upload e dowload, ...) si fa uso della libreria Jakarta Commons Net (pagina di riferimento http://jakarta.apache.org/commons/net/). Si farà uso della classe di utilità Commander creata appositamente per questo progetto (per i dettagli far riferimento al progetto allegato). Tale classe avrà un metodo login per ottenere una connessione e il metodo list che restituisce la lista dei file (vettore di oggetti di tipo FTPFile):

```
public class loginAction extends ActionSupport{
 private login myLogin;
 private FTPFile[] listaFile;
 //...
 public String execute() {
  Commander cmd = null;
  try {
   cmd = new Commander(myLogin);
   listaFile = cmd.list();
  } catch (Exception e) {
   return ERROR;
  }finally{
   try{
    cmd.close();
   }catch(Exception ex){ ; }
  }
 return SUCCESS; }
}
```

Se c'è stato un errore, allora verrà restituita la costante ERROR e, per com'è stato costruito il file xwork.xml, verrà invocata la pagina login.jsp, altrimenti verrà invocata la pagina lista.jsp. Quest'ultima dovrà reperire il vettore listaFile e stampare le informazioni volute. Siccome i metodi set"Nome" sono usati per passare da parametri della rechiesta ai valori dei bean, è naturale pensare che i metodi get"Nome" saranno utilizzati per reperire i risultati dai bean. Infatti è usata proprio questa convenzione. Ecco, pertanto, il metodo da aggiungere a loginAction:

```
public FTPFile[] getListaFile() {
 return listaFile;
}
```

A livello di jsp si vuole, presumibilmente, scorrere il vettore e stampare, per esempio, il nome e la dimensione di ciascun file; si può ricorrere ancora una volta alla libreria di tag predefinita di WebWork e, in particolar, al tag iterator che permette di che scorre gli elementi di una qualsiasi lista o vettore:

```
<ww:iterator value="listaFile" id="vettore">
 <ww:property value="name" />, <ww:property
                                    value="size" /><br/>
 </ww:iterator>
```

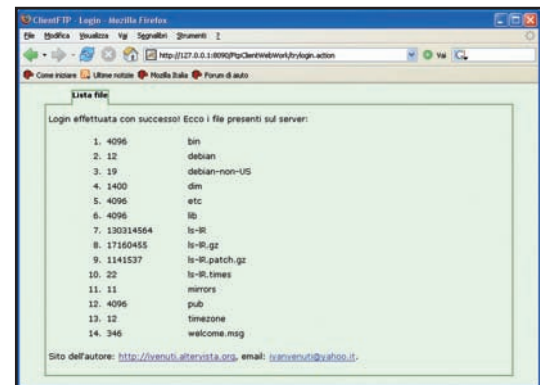



Fig. 3: ***Il disegno di una funzione ad una variabile***

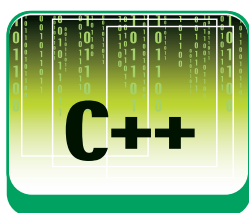
In questo caso, all'interno di iterator, ciascuna property si riferisce ad un metodo get dell'oggetto su cui si sta iterando (iterando su listaFile, la proprietà "name" farà riferimento a listaFile.getName(); in maniera simile size farà riferimento a listaFile.getSize()). In **Figura 3** un esempio dell'applicazione finale.

CONCLUSIONI

WebWork, oltre alle funzionalità descritte, permette di utilizzare numerosi altri tipi di interceptors, tag per la visualizzazione dei dati e un qualsiasi container che realizzi il pattern Inversion of Control (IoC). Ovviamente accanto a questa nutrita serie di oggetti e tag è possibile crearne di nuovi adatti per scopi specifici e personalizzati. Tutto questo permette di usare WebWork per adattarlo alle proprie esigenze e non viceversa. Il fatto che presto verrà rilasciato con il nome di Struts 2 è garanzia di continuità, supporto a lungo termine e, cosa fondamentale, sempre più documentazione e interesse verso questo eccellente framework.

Ivan Venuti

CONVIVENZA FRA C++ NATIVO E .NET

SE SEI UN PROGRAMMATORE C++ È PROBABILE CHE AVRAI SCRITTO PARECCHIO CODICE OTTIMIZZATO PER LE TUE ESIGENZE CONTROLLANDO FINO ALL'ULTIMO BIT. COME PUOI RIUTILIZZARE LO STESSO CODICE IN .NET? TE LO SPIEGHIAMO IN QUESTO ARTICOLO...

In questo articolo ci occuperemo di come "riciclare", o meglio riutilizzare, codice originariamente scritto in C++ nella piattaforma .NET. Come caso d'uso pratico prenderemo in esame il calcolo del CRC32 di qualsiasi file. L'algoritmo CRC32 è un algoritmo di hashing utilizzato in numerosi programmi che utilizziamo quotidianamente: dal celebre winzip all'altrettanto famoso programma peer to peer eMule (questo solo per citare i più conosciuti). Avremo comunque modo di parlare più diffusamente degli algoritmi di hashing, in particolare del crc32, nel proseguo dell'articolo stesso.
Prima di affrontare questi aspetti mi sembra opportuno, sotto certi aspetti anche doveroso, spendere qualche parola sul C/C++ in modo tale da partire poi con un background condiviso, soprattutto per quei programmatori (spesso i più giovani) che non hanno avuto modo di sviluppare quel sentimento di odio/amore nei confronti di questo linguaggio.

LINGUAGGI COMPILATI, INTERPRETATI E VIE DI MEZZO

Ormai la maggior parte degli IDE utilizzati per sviluppare codice, al di là del linguaggio utilizzato, rende (fortunatamente) quasi completamente trasparente tutto il processo necessario a passare dal nostro codice sorgente che adotta una sintassi molto vicina ad un "linguaggio umano" alle istruzioni che lo fanno girare sulla "cpu" su cui si basano i nostri computer.
Cerchiamo quindi di fare una brevissima panoramica sui modi di passare dal nostro sorgente a ciò che la macchina esegue realmente.
Partiamo con i linguaggi compilati, della quale famiglia fa parte il C/C++, anzi possiamo dire che ne sia il padre-padrone.
I linguaggi compilati vengono tradotti in file eseguibili (che sulle piattaforma Microsoft sono tipicamente gli exe) di codice macchina binario da uno speciale programma chiamato (ovviamente) compilatore. Una volta che il codice binario è stato generato potete eseguirlo direttamente senza più guardare al codice sorgente. Infatti la maggior parte del software viene distribuita come binari compilati a partire da codice che non vedete.
Un linguaggio interpretato di contro necessita appunto di un interprete che legge il codice sorgente e lo traduce al volo in calcoli e chiamate di sistema. È ovvio che il sorgente debba essere reinterpretato (e l'interprete deve essere presente) ogni volta che il codice viene eseguito. A sua volta però l'interprete dovrà essere un programma compilato (in linea teorica si potrebbe immaginare anche un interprete interpretato a sua volta da un altro interprete, ma in realtà ciò non avviene mai). Molti linguaggi utilizzati per il web rientrano in questa categoria, sia quelli lato server come il PHP sia quelli lato client come JavaScript.
Storicamente i linguaggi interpretati sono nati successivamente a quelli compilati per diverse ragioni. La prima e più ovvia è quella che, come detto, un programma interpretato a bisogno di un interprete che deve essere compilato. La seconda invece è un po' più sottile: non a caso sono stati citati due linguaggi usati per il web.
Essi infatti dovevano rispondere a due esigenze imprescindibili per un ambiente come il web: sicurezza e essere indipendenti dalla piattaforma su cui vengono fatti girare.
In questo caso infatti, non essendo più necessario ottenere un file binario, ossia l'eseguibile, dai sorgenti sarà possibile far girare il programma senza preoccuparsi della piattaforma su cui quest'ultimo è stato lanciato. Allo stesso tempo il fatto che il programma interpretato non comunichi direttamente con il sistema

operativo sottostante ma che tutta avvenga tramite “l'intermediazione” di un interprete garantisce che programmi mal funzionanti o volutamente maliziosi danneggino l'intero sistema (vedi BOX1).
Come potete ben immaginare queste nuove ed utili caratteristiche si pagano generalmente in termini di performance (non ci sogneremo mai di scrivere ad esempio un programma real-time in PHP!).
Un famoso motto latino recita: “in medio stat virtus” ed alcuni volte questo è proprio vero. A metà strada fra queste due opzioni esistono i linguaggi come Java o quelli del framework .NET (C#, Visual Basic .NET, Visual C++ Managed).
Per questo tipo di linguaggi infatti è presente sia un compilatore che una specie di interprete.
Il compito del compilatore è questa volta non è quello di generare un file binario direttamente eseguibile dal sistema operativo, bensì un binario che venga eseguito dall'interprete. Nel caso di Java tale binario si chiama bytecode e l'interprete è la Virtual Machine (JVM), mentre nel caso .NET il binario si chiama assembly (nulla o quasi a che vedere con Assembler) e l'interprete è il Common Language Runtime (CLR). È proprio di questo secondo caso che andremo ad occuparci.

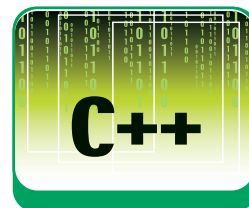
C++ UNMANAGED E C++ MANAGED

All'inizio dell'articolo abbiamo affermato che i moderni IDE rendono quasi trasparente il processo di trasformazione che permette di trasformare i sorgenti ad un generico eseguibile (sia che esso sia un compilato od un interpretato). Questa affermazione è vera ma solo fino ad un certo punto.
Se infatti è vero che solitamente è l'IDE a farsi carico di comunicare con l'eventuale compilatore od interprete, altra cosa sono le possibilità messe a disposizione e le accortezze da adottare quando si programma con un linguaggio che girerà direttamente su di un microprocessore (compilato) rispetto ad uno che verrà eseguito da un interprete (l'esempio del BOX1 ne è già un primo esempio).
Una delle croci e delizia dei programmatori di C/C++ sono i puntatori, cioè è possibile ottenere l'indirizzo di memoria di dove è allocata una variabile od un oggetto, cosa invece non concessa da altri linguaggi come C# o VB.NET.
Su come trattare opportunamente i puntatori sono state scritte pagine e pagine e non è questa la sede per affrontare diffusamente tale questione, l'unica cosa di cui ci interessa farci un'idea è solo la differenza che sussiste tra l'allocazione esplicita di un oggetto nell'heap rispetto alla presenza di una garbage collector.
La parola chiave new assume in C++ un significato abbastanza diverso rispetto al C# o VB.NET.
In C# ad esempio la creazione di un oggetto è semplicemente:

```
public void method(...)
{
        ..fa qualcosa...
        ObjectX obj = new ObjectX();
        ....
        obj.m();
}
```

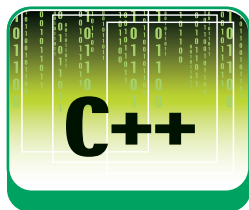
Quando method restituirà il controllo al chiamante, di obj non ci sarà più traccia. In realtà quello che succede dietro le quinte è che periodicamente il garbage collector controllerà quali degli oggetti allocati non sono più referenziati e provvederà così a liberare lo spazio di memoria relativo.
In C++ la cosa si fa un po' più complicata:

```
public void method(...)
{
        ..fa qualcosa...
        ObjectX * obj = new ObjectX();
        ....
        obj->m();
}
```

l'operatore new in quest'ultimo caso restitui-

INTERPRETI E COMPILATORI

Per spiegare meglio quanto in realtà possa essere diverso programmare con un linguaggio compilato come C/C++ rispetto ad uno interpretato (anche se passa per un bytecode) prenderemo ad esempio uno stralcio di codice comunissimo e apparentemente innocuo, il riempimento di un array:

```
for(int i=0;  i<N; i++)
        array[i]=0;
```

Vista la semplicità di quanto scritto sopra tale codice può essere portato sia su C++ che su C# o Java, però il comportamento a run-time dello stesso può essere profondamente diverso negli ultimi due casi rispetto al primo. Infatti in C# o Java la VM controlla che l'indice sia compreso nel range dell'array stesso, in caso contrario verrà sollevata un'eccezione che ci avverte di ciò. Nel caso del C++ non c'è una VM che si occupa di ciò, non solo, ma l'informazione di quanto sia lungo l'array stesso non è proprio presente. Così si può tranquillamente andare oltre la memoria riservata nell'heap e scrivere porzioni di memoria contigue senza neanche rendersene conto!

sce non l'oggetto bensì il puntatore ad esso, o meglio ancora l'indirizzo della memoria heap dove esso è allocato. Una rappresentazione grafica di quanto detto è mostrata in figura 1.
Benché il codice che abbiamo tradotto in C++ segua in maniera pedissequa il precedente, in questo caso c'è qualcosa che non va; proprio come quando si traduce da l'italiano all'inglese tutti sappiamo che non si può tradurre parola per parola, analogamente dovremmo comportarci col C++.
Non essendo presente la garbage collector, al momento della restituzione del controllo al chiamante nessuno si occuperà di liberare (o meglio deallocare) la memoria; per questo motivo esiste un operatore simmetrico a new che si chiama delete. Infatti al contrario di new che si occupa di riservare memoria per l'oggetto da allocare, delete si occuperà di deallocare la memoria precedentemente riservata. Il codice corretto sarà quindi il seguente:

```
public void method(...)
{
        ..fa qualcosa...
        ObjectX * obj = new ObjectX();
        ....
        obj->m();
        delete obj;
}
```

Se così non facessimo andremmo incontro al così detto memory leakage. Infatti nel caso di una singola chiamata potremmo anche accettare di "sprecare" un po' di memoria, altra storia sarebbe il caso ad esempio di una serie di chiamate ricorsive (si pensi a cosa accadrebbe per una serie di Fibonacci!).

Fig. 1: Esempio di come viene gestita la memoria heap a basso livello

Tutto questo discorso è stato fatto per porci una domanda cruciale: com'è possibile integrare questo tipo di programmazione con un linguaggio che prevede la presenza di una garbage collector?
In prima istanza possiamo affermare che gran parte delle problematiche ce le hanno già risolte i ragazzi di casa Microsoft.
Visual Studio infatti permette di programmare in tutti i linguaggi previsti da .NET e tra questi c'è anche il C++. La doppia esigenza di non perdere i "vecchi programmatori di C/C++" e quella di permettere una "transizione indolore" alla nuova piattaforma .NET ha portato a progettare un'architettura sintetizzabile in figura 2.
Invece di spendere un ulteriore fiume di parole su quest'aspetto, presentiamo un esempio concreto ed utile per comprendere bene come sia possibile effettuare tale integrazione.
Teniamo quindi a mente quanto esposto fino ad adesso e occupiamoci per un istante di descrivere cosa siano gli algoritmi di hashing.

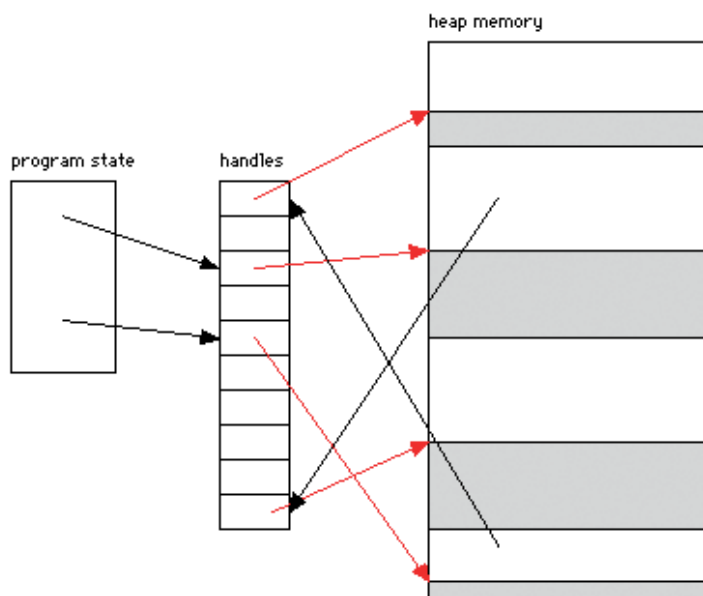
ALGORITMI HASHING E CRC32

Un algoritmo di "Hash" elabora qualunque quantità di bit (in informatica si dice "digerisce tutto ciò che gli viene dato in pasto") e restituisce una stringa di bit di lunghezza fissa e predefinita. Si tratta di una famiglia di algoritmi che soddisfa questi requisiti:

1)L'algoritmo ritorna una stringa di numeri e lettere a partire dal documento di qualsiasi dimensione che viene entrato. Tale stringa è detta Digest.

2)La stringa è univoca per ogni sequenza di byte e ne è un'identificatore. È per questo che l'algoritmo è utilizzabile per la firma digitale.

3)L'algoritmo non è invertibile, ossia non è possibile ricostruire il documento originale a partire dalla stringa che viene ritornata in output.

Il bello di questi algoritmi è quindi la possibilità di avere digest diversi anche cambiando un solo bit dell'ingresso. Tra le tante applicazioni che questo tipo di algoritmi ha, c'è

anche quella del controllo dell'integrità di un file. In particolare, per realizzare ciò, utilizzeremo l'algoritmo CRC32, appartenente alla famiglia degli hashing.

NON REINVENTARE MAI LA RUOTA!

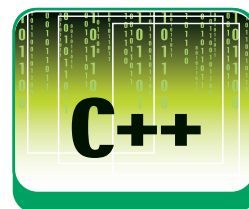
Quando per la prima volta mi accinsi a utilizzare tale algoritmo rimasi sorpreso alquanto nel constatare che il framework .NET non lo prevedesse già nel namespace System.Security.Cryptography (assieme a tutti gli altri MD5, SHA1 etc...)
A questo punto avevo di fronte a me due strade alternative: la prima era quella di ristudiare l'algoritmo ed implementarlo ex-novo nel linguaggio che intendevo utilizzare, l'altra era quella di chiedere a qualche amico-collega se già non avesse una classe che facesse al caso mio. Come potete facilmente intuire la strada maestra da seguire era indubbiamente la seconda. Ciò che ho fatto prima di è stato riprendere in mano la classe scritta da Bartosz Milewski e messa a disposizione all'indirizzo http://www.relisoft.com, di cui riporto l'interfaccia qui sotto (l'implementazione la trovate nel cpp allegato al CD).

```
class CCrc32
{
  public:
    enum EPolynomials
    {
      polyStandard = 0x04C11DB7,
      polyStandardReversed = 0xEDB88320
    };

  public:
    CCrc32(DWORD dwPoly);
    void PutByte(BYTE byByte);
    void PutBuffer(BYTE* pbyBuf, DWORD
                                    dwBufSize);
    DWORD Done();

  protected:
    DWORD m_dwPoly;
   // really 33-bits key,
    // counting implicit 1 top-bit
    DWORD m_dwRegister;
    DWORD m_adwTable[256];
};
```

I tipi DWORD e BYTE sono due tipi definiti così:

```
typedef unsigned long DWORD;
```

```
typedef unsigned char BYTE;
```

ossia rispettivamente 32 e 8 bit.

COSTRUTTORI E DISTRUTTORI

Ora torniamo a quanto detto in precedenza sulla differenza di allocazione di memoria tra il C++ Managed (quello che gira sotto la CLR) ed il C++ Unmanaged (quello compilato) come mostrato in figura 2.
Diamo subito una procedura operativa che si può compiere ogni qualvolta si incontra un problema del genere, ossia integrare codice C++ nel framework .NET.
Per prima cosa creiamo una nuova solution di tipo Visual C++ Class Library che chiameremo CRC32Wrap (il perché di questo nome sarà più chiaro tra poco). A tale solution aggiungiamo poi il progetto dove è contenuto il codice nativo, che nel mio caso ho chiamato, con poca fantasia, CRC32Native. Arrivati a questo punto dovreste avere un solution explorer simile a quello riportato in figura 3.
Quello che ci rimane ora da fare è implementare un wrapper che funga da "bridge" tra il mondo .NET e quello C++ nativo. La parte che segue rappresenta uno standard applicabile a tutti i problemi di questo tipo:

1 Creiamo una classe wrapper che contenga al proprio interno un puntatore all'oggetto di cui tale classe è wrapper:

```
public ref class CRC32
      {
      ................
      private:
                  CCrc32 * m_poCRC32;
```

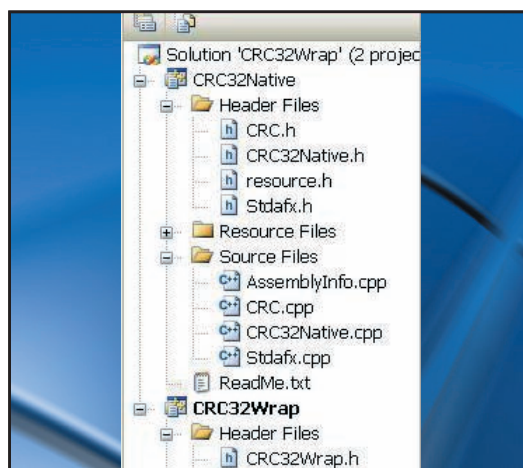
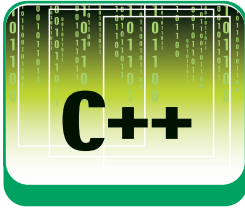



Fig. 3: ***Struttura della solution contenente sia il progetto nativo che il wrapper***

2 Definire i costruttori ed i distruttori in modo tale che la garbage collector sappia trattare correttamente la classe in questione. Questa fase è in fondo più facile a farsi che a dirsi:

```
public:
        CRC32(System::UInt32 poly) :
        m_poCRC32(new CCrc32(poly)){}
        ~CRC32()    {
                delete  m_poCRC32;
        }
protected:
        !CRC32()    { delete
                m_poCRC32; }
```

Il distruttore protetto, che inizia con il simbolo !, è quello che verrà invocato dalla garbage collector quando deciderà di liberare la heap memory dagli oggetti non più referenziati.
Un'altra cosa da notare, che sarà meglio puntualizzata nel punto 3, è la necessità di "esporre" dei tipi che siano previsti dalla CLR al posto dei typedef usati nel codice nativo, in questo caso DWORD è divenuto Uint32, ossia un intero unsigned.

3 Fornire la classe wrapper di tutti i metodi necessari affinché possa essere correttamente ed efficentemente usata da qualsiasi altro linguaggio .NET. Come già notato nel punto 2, ciò implica ad esempio che tale classe dovrà esporre solo i tipi previsti dal framework stesso. Un'altra "best practice" è quella di evitare di aggiungere logica alla classe wrapper. In altre parole è vivamente sconsigliato aggiungere ulteriori funzionalità alla classe wrapper rispetto a quella nativa. Benché nulla vieti di poter fare il contrario, verrebbero meno le motivazioni principale che ci hanno portato a utilizzare questa tecnica, ossia riutilizzare codice già esistente!
Nel nostro esempio tale principio si traduce:

```
void put(BYTE byByte){
                list.Add(byByte);
};

System::UInt32
 IoProgrammo::Galeazzi::Andrea::CRC32::Done()
{
        BYTE * pbyBuffer =
                        new BYTE[list.Count];
        for(int i=0; i<this->list.Count; i++)
        {
                pbyBuffer[i] = this->list[i];
        }

        this->m_poCRC32->PutBuffer
                        (pbyBuffer,list.Count);
        System::Int32 result =
                this->m_poCRC32->Done();
        delete[] pbyBuffer;
        return result;
}
```

La nostra classe wrap prevede due soli metodi public: put e done.
Scendendo più nel dettaglio abbiamo scelto di effettuare un wrap con una logica di tipo "batch", ossia invece di richiamare ogni volta la classe nativa abbiamo accumulato i byte per il buffer necessario all'algoritmo CRC32 in un semplice contenitore generic previsto dal framework .NET.
Solamente al momento del calcolo effettivo dell'hashing, ossia all'invocazione del metodo Done(), viene realmente creato tale buffer di byte, "flushato" nella classe nativa ed invocato il proprio metodo Done().
Va inoltre notato come sia stato necessario eliminare tale buffer prima della restituzione del controllo al chiamante per evitare il problema di "memory leakage" discusso in precedenza.

Fig. 2: *rappresentazione dell'architettura C++ manged- unmanaged*

UTILIZZIAMO LE DLL DOVE VOGLIAMO!

Finalmente ora possiamo utilizzare la nostra libreria in qualsiasi applicazione .NET che vogliamo. Tenendo presente la struttura della nostra solution riportata in **figura 2**, le proprietà di compilazione devono essere simili a

quelle riportate in figura 4.
Arrivati a questo punto possiamo riutilizzare la DLL così creata semplicemente facendo un "Add Reference" nella applicazione che ne necessita.
Nel nostro caso svilupperemo una GUI che ci permetterà di riconoscere se un file è stato corrotto o meno.
Dato che siamo giunti quasi alla fine di quest'articolo, di seguito ci concentreremo sull'integrazione di tale libreria nel codice Visual Basic .NET che implementa la GUI stessa. Per questa "prima release" ci accontenteremo di catturare un evento di selezione su un componente per il browsing del file system e visualizzare l'estratto hashing CRC32 in formato esadecimale.
Una volta aggiunta la reference alla DLL creata precedentemente, il codice necessario a compiere il task sopra descritto diviene veramente semplice:

```
Private Sub treeView_AfterSelect(

  ByVal sender As Object, ByVal e As
        System.Windows.Forms.TreeViewEventArgs)
              Handles  treeView.AfterSelect
    If IO.File.Exists(

        treeView.SelectedNode.FullPath) Then
        Dim crc32 As New
           IoProgrammo.Galeazzi.Andrea.CRC32(
                              IoProgrammo.
              Galeazzi.Andrea.CRC32.directPoly)
        Dim contents As Byte() =
                         IO.File.ReadAllBytes
              (treeView.SelectedNode.FullPath)
        Dim myByte As Byte
        For Each myByte In contents
           crc32.put(myByte)
        Next
        TextBox1.Text =
              crc32.Done.ToString("x")
     End If

  End Sub
```

Oltre al controllo per verificare se il file veramente esiste, non abbiamo fatto altro che:

1 – Leggere tutti i byte del file
2 - Istanziare la classe CRC32
3 – Riempire il buffer con i byte del file
4 – Chiamare il metodo Done per ottenere l'estratto hash.

Una prima schermata di tale applicazione la trovate in figura 5.

CONCLUSIONI

In quest'articolo si è voluto rendere noto una possibilità che il framework .NET mette a disposizione e che il più delle volte viene completamente ignorato. In questo modo abbiamo ottenuto l'integrazione fra codice nativo C++ e codice .NET. Come esempio abbiamo trattato degli algoritmi di hashing ed in particolare del CRC32. Nel prossimo articolo amplieremo ulteriormente le potenzialità di tale applicazione in modo tale da avere un vero e proprio "controllore" del nostro file system.

Andrea Galeazzi

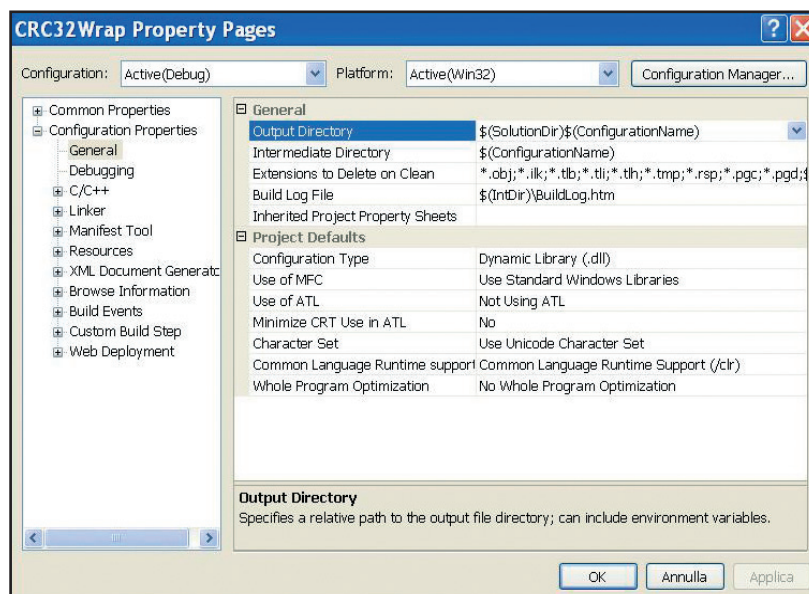
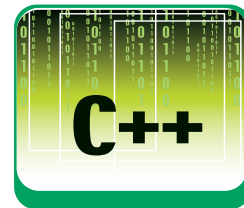



Fig. 4: Schermate delle proprietà generali del project wrap

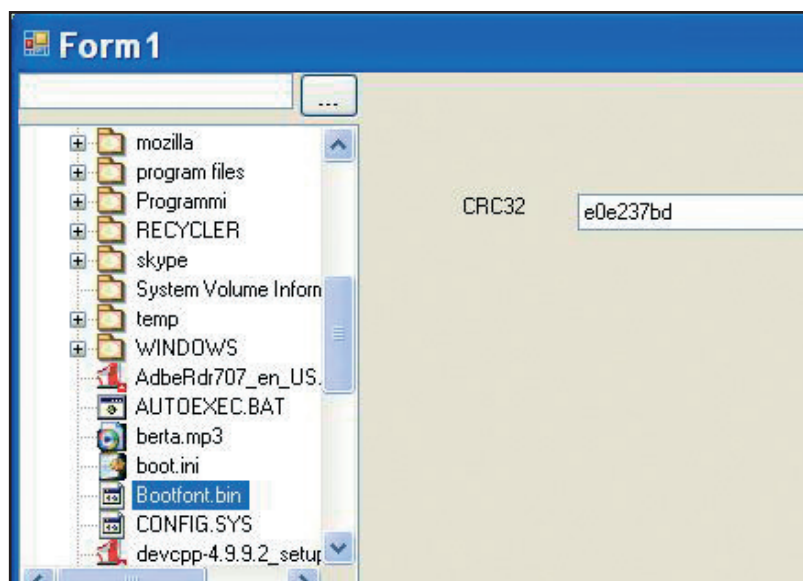



Fig. 5: Schermata principale dell'applicazione

DICIAMO ADDIO AL CODICE SPAGHETTI

IMPARIAMO A USARE IL TEMPLATE METHOD, UNO DEI PATTERN PIÙ DIFFUSI E CHE CI CONSENTE DI SCRIVERE CODICE FACILMENTE MANUTENIBILE. SCOPRIREMO CHE ALCUNI COSTRUTTI DEI LINGUAGGI NON SONO SOLO ORNAMENTI

Sono passati pochi giorni da quando abbiamo consegnato l'ultima versione del nostro sistema di prenotazioni aeree, e già cominciano i problemi. Pare che il sistema abbia venduto molti più posti di quelli disponibili. Un centinaio di passeggeri inferociti sono rimasti bloccati in aeroporto e hanno preso in ostaggio il personale di volo. La polizia ha circondato l'edificio e sta cercando di convincere i passeggeri a non uccidere gli ostaggi. La compagnia di volo nostra cliente non è particolarmente contenta, e ci ha già inviato minacciose ingiunzioni legali. Ed è solo lunedì. Ci aspetta dunque una stressante giornata passata a correggere bug. Il problema sta nel fatto che, come vedremo, abbiamo strutturato le classi lasciandogli "troppa libertà" e mettendoci nelle condizioni di sbagliare. Esistono modi per evitare questo problema, e in questo articolo ne vedremo uno: il pattern Template Method.

Il codice è scritto in Python, ma non preoccupatevi se non lo conoscete – non è roba complicata. I clienti che vogliono prenotare un posto sono oggetti della classe Customer:

```
class Customer:
    def grant(self, services):
        self.services = services
    def assign(self, seat):
        self.seat = seat
    def __str__(self):
        return "Posto: %d  Servizi: %s" % (self.seat,
                                          self.services)
```

Questa classe contiene tre metodi: il primo assegna un elenco di servizi al cliente; il secondo gli assegna un numero di posto; il terzo è un metodo speciale che viene chiamato automaticamente quando si stampa un oggetto (è simile al toString() di Java), e in questo caso stampa una stringa contenente i servizi e il posto assegnati al cliente. I servizi e il posto vengono conservati in due campi, self.services e self.seat. Python è un linguaggio dinamico, quindi non dobbiamo dichiarare i campi: basta assegnargli un valore, e appariranno "automagicamente".

Questi *Customer* sono solo dei semplici contenitori di dati. Per trovare un po' di logica dobbiamo guardare nella gerarchia delle classi che definiscono i voli. Abbiamo una classe base e due sottoclassi, una per i voli Low Cost e una per quelli di lusso. Ecco la classe base:

```
class Flight:
    def __init__(self, seats):
        self.free_seats = seats
    def check_seats(self):
        if self.free_seats == 0:
            raise "Overbooking!"
```

Nel metodo *__init__*, che è l'equivalente Python di un costruttore, viene passato (oltre al solito self) il numero dei posti liberi quando l'aereo è vuoto. Questo numero viene conservato nel campo *free_seats*. La classe ha anche un metodo check_seats che verifica se i posti sono finiti, e in questo caso lancia un'eccezione con il comando raise. L'idea è che ciascuna sottoclasse deve ricordarsi di chiamare questo metodo durante una prenotazione per controllare che ci siano ancora posti liberi. Ad esempio, ecco un volo a basso costo:

```
class LowCostFlight(Flight):
    def book(self, customer):
        self.check_seats()
        customer.assign(self.free_seats)
        self.free_seats -= 1
        customer.grant(["Tozzo di pane secco", "Acqua
                                          di fonte"])
```

LowCostFlight eredita da *Flight*. Il metodo per prenotare un posto si chiama *book* ("to book" in inglese significa "prenotare"). Questo metodo riceve un cliente (customer), e per prima cosa controlla che i posti liberi non siano esau-

riti. Se non saltano fuori eccezioni, assegna al cliente un numero di posto corrispondente al numero di posti liberi e decrementa il numero di posti liberi. Ad esempio, se ci sono dieci posti liberi, al cliente viene assegnato il posto numero 10 e il numero di posti liberi diventa 9. Poi il metodo assegna al cliente la lista dei servizi cui ha diritto per questo tipo di volo (le parentesi quadre definiscono una lista). Possiamo scrivere una semplice funzione per testare questo codice:

```
def booking_test(flight, number_of_bookings):
    for i in range(number_of_bookings):
        c = Customer()
        flight.book(c)
        print c
```

Questa funzione accetta un volo e un numero di prenotazioni. Il codice è un ciclo che va da uno al numero delle prenotazioni (questo è il significato di for e range). Ad ogni passo del ciclo, il test crea un nuovo Customer, gli fa prenotare il volo e ne stampa lo stato. Cosa succede se prenoto tre posti su un aereo che ne ha solo due?

```
booking_test(LowCostFlight(2), 3)
```

Il risultato è che il sistema assegna il posto 2 al primo cliente, il posto 1 al secondo, e poi (dato che i posti sono finiti) lancia un'eccezione:

```
Posto: 2  Servizi: ['Tozzo di pane secco', 'Acqua di
                                              fonte']
Posto: 1  Servizi: ['Tozzo di pane secco', 'Acqua di
                                              fonte']
Traceback (most recent call last):
  File "flights1.py", line 40, in ?
    booking_test(LowCostFlight(2), 3) # Genera
                                         un'eccezione
  File "flights1.py", line 37, in booking_test
    flight.book(c)
  File "flights1.py", line 23, in book
    self.check_seats()
  File "flights1.py", line 19, in check_seats
    raise "Overbooking!"
Overbooking!
```

Ora andiamo a vedere cosa succede nella seconda sottoclasse di *Flight*:

```
class LuxuryFlight(Flight):
    def book(self, customer):
        customer.assign(self.free_seats)
        self.free_seats -= 1
        customer.grant(["Ostriche al tartufo",
          "Champagne a fiumi", "Massaggio ai piedi"] )
```

Il codice di questo volo di lusso è quasi identico a quello del volo a basso costo. Ci sono solo due differenze: la prima è che la lista dei servizi è diversa, e la seconda... Ehi, un momento! Ecco il nostro bug! Chi ha scritto il codice di questa classe ha dimenticato di chiamare il metodo *check_seats*, quindi il volo di lusso non genera alcun errore se riceve più prenotazioni del numero di posti disponibili.

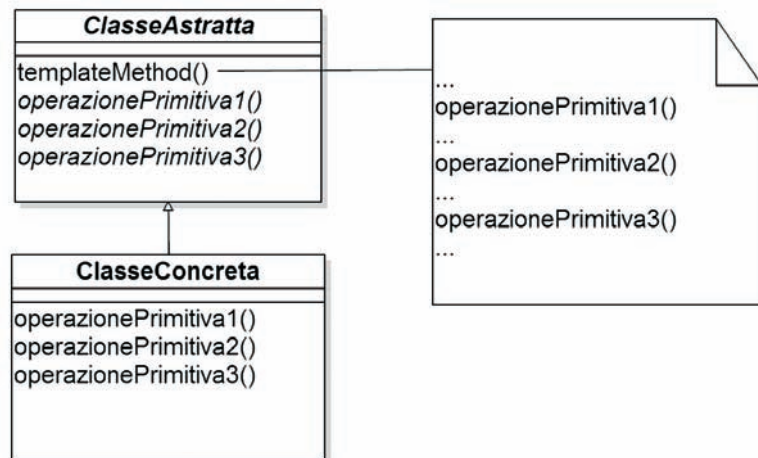



Fig. 1: ***Una rappresentazione schematica del template method***

Verifichiamolo:

```
booking_test(LuxuryFlight(2), 3)
```

Il risultato è:

```
Posto: 2  Servizi: ['Ostriche al tartufo', 'Champagne
                        a fiumi', 'Massaggio ai piedi']
Posto: 1  Servizi: ['Ostriche al tartufo', 'Champagne
                        a fiumi', 'Massaggio ai piedi']
Posto: 0  Servizi: ['Ostriche al tartufo', 'Champagne
                        a fiumi', 'Massaggio ai piedi']
Posto: -1  Servizi: ['Ostriche al tartufo',
             'Champagne a fiumi', 'Massaggio ai piedi']
```

Sul nostro aereo non esiste un posto zero, e tanto meno un posto -1. Non c'è da stupirsi se molti viaggiatori sono rimasti a terra. Per cor-

DELIMITARE UN BLOCCO

Chi non conosce Python potrebbe stupirsi per la mancanza di parentesi graffe. Per delimitare un blocco di codice basta indentarlo. Ogni volta che referenziate un metodo o un campo dovete usare un riferimento all'oggetto corrente (self). Tenete anche presente che il primo parametro di tutti i metodi (che anche in questo caso, per convenzione, si chiama self) viene assegnato automaticamente dal sistema quando il metodo viene chiamato. Questo vuol dire che potete chiamare i metodo assign e grant come se avessero un solo parametro

reggere il problema ci basterebbe aggiungere la chiamata mancante al metodo check_seats. Ma è meglio evitare che questo incidente possa capitare ancora in futuro, quando aggiungeremo altri tipi di volo. Visto che ci siamo, possiamo anche ripulire il codice dalle righe duplicate nelle sottoclassi. Cominciamo con una versione più intelligente di Flight:

```
class Flight:
    def __init__(self, seats):
        self.free_seats = seats
    def assign_seat(self, customer):
        customer.assign(self.free_seats)
        self.free_seats -= 1
    def book(self, customer):
        if self.free_seats == 0:
            raise "Overbooking!"
        self.assign_seat(customer)
        self.grant_services(customer)
```

Ora tutta la gestione del campo *free_seats* avviene qui dentro. Le sottoclassi non hanno più bisogno di modificare questo campo, e nemmeno di vederlo. L'intero metodo book è stato spostato nella superclasse, e fa tutto quello che serve per prenotare un volo - escluse le operazioni specifiche delle sottoclassi. Per prima cosa controlla che ci siano posti liberi. Poi assegna il posto al cliente tramite un nuovo metodo di nome *assign_seat*, che include le due righe che prima erano duplicate nelle sottoclassi. Infine chiama il metodo *grant_services* che assegna i servizi al cliente. Quest'ultimo metodo è interessante, perché non è definito nella classe *Flight*. Devono essere le sottoclassi a definirlo:

```
class LowCostFlight(Flight):
    def grant_services(self, customer):
        customer.grant(["Tozzo di pane secco", "Acqua
                                              di fonte"])
class LuxuryFlight(Flight):
    def grant_services(self, customer):
        customer.grant(["Ostriche al tartufo",
          "Champagne a fiumi", "Massaggio ai piedi"] )
```

Ora le responsabilità sono distribuite bene: tutta la logica comune è nella superclasse, e le sottoclassi contengono le operazioni che cambiano tra diversi tipi di volo. Quando scriviamo una nuova sottoclasse non dobbiamo più duplicare il codice, ma soprattutto non rischiamo di dimenticare qualche operazione importante. Al massimo possiamo dimenticare di definire *grant_services()*, ma in questo caso il programma fallisce subito con un chiaro messaggio di errore. Se vogliamo, possiamo anche ridefinire *assign_seat()* in una sottoclasse per cambiare il modo in cui vengono assegnati i posti (ad esempio in alcuni voli potremmo volerli assegnare dal primo all'ultimo piuttosto che dall'ultimo al primo).

E per continuare con le buone notizie, il telegiornale dice che tutti gli ostaggi sono stati liberati in cambio di un volo speciale per le Maldive. La settimana volge al meglio!

IL BELLO DEI PATTERN

Il concetto di "pattern" è nato nel campo dell'architettura – quella degli edifici, non quella del software. Negli anni '60, un architetto di nome Christopher Alexander notò che tutte le case, le piazze e i palazzi che "funzionavano bene" avevano caratteristiche comuni. Queste caratteristiche non si ripetevano quasi mai in modo perfettamente identico, ma le si riconosceva negli case tradizionali sudamericane come nelle capanne africane e nelle moderne abitazioni europee. Alexander chiamò queste soluzioni ricorrenti "pattern". Negli anni '90, alcuni programmatori lessero il libro di Alexander e ne furono illuminati. In nessun campo come nell'ingegneria del software si risolvono problemi sempre simili, ma mai perfettamente uguali. Nel 1995 uscì "Design Patterns", che tuttora è uno dei libri che non possono mancare nella biblioteca di un informatico. Il libro era un catalogo di ventiquattro pattern comuni, incluso Factory Method. Anche se il libro non è di facile lettura, vale decisamente la pena di affrontarlo. All'interno vengono affrontate tutta una serie di problematiche che trovano soluzione per mezzo di modelli prestabiliti. Nel tempo le proposte segnalate come "Pattern" sono diventate un dato acquisito fino a diventare una regola in fatto di programmazione. Ovviamente tutto va adattato alle proprie esigenze ma modellare i propri software sulla base di un pattern sicuramente è una garanzia di riuscita del progetto

IL PATTERN TEMPLATE METHOD

Il metodo *book* della classe *Flight* è un esempio di *Template Method*. L'idea fondamentale di questo pattern è: la superclasse contiene un metodo che definisce un algoritmo, e i dettagli di questo algoritmo sono delegati ad altri metodi. Questi ultimi si chiamano a volte *operazioni primitive*, e vengono definiti (o ridefiniti) dalle sottoclassi per essere chiamati polimorficamente. Di solito la superclasse è astratta, cioè non è fatta per essere istanziata, ma solo per essere ereditata – quindi è accettabile che contenga degli "spazi vuoti" che le sottoclassi possono riempire.

Il Template Method è uno dei pattern più fondamentali della programmazione a oggetti. In effetti è probabile che lo abbiate già usato molte volte senza nemmeno pensare che aves-

se un nome.
Questo pattern è anche alla base del concetto di *framework*. Un framework è diverso da una libreria perché le responsabilità sono opposte: quando usate una libreria, siete voi che chiamate il codice della libreria; quando usate un framework, è lui che chiama il vostro codice. Un framework è un ambiente di programmazione quasi pronto per essere usato, con qualche spazio vuoto. Il punto di contatto tra un framework e l'applicazione è tipicamente un Template Method in una classe del framework. Il programmatore può ereditare da questa classe e "riempire gli spazi" scrivendo il codice delle primitive. La stessa idea si ritrova, in modo più astratto, nel concetto di *plugin*.
Questo principio per cui l'ambiente chiama il codice dell'applicazione, e non viceversa, esisteva anche prima della programmazione a oggetti, e si chiamava *callback*. Un Template Method è semplicemente un modo per scrivere dei callback sfruttando il polimorfismo. Lo stesso principio è anche noto come *inversione di controllo*, o come il *principio di Hollywood*: "Non ci chiami lei, la chiameremo noi".

CONCLUSIONI

Grazie al Template Method possiamo ridurre il codice duplicato e scrivere classi da cui si può ereditare facilmente e senza paura di sbagliare. E' uno dei pattern più usati, ma anche uno dei più semplici. Questo approccio inoltre ci mette nelle condizioni di sbagliare di meno e ottenere codice facilmente manutenibile. Alla base c'è l'idea della modularità e un uso abbastanza intelligente delle classi astratte. Utilizzare questo genere di template è un must anche per i progetti più semplice.
Ma ci aspettano pattern più sofisticati ed eleganti. Arrivederci al prossimo numero di IoProgrammo!

Paolo Perrotta

QUANDO C'È UN TIPO DI MEZZO

Cosa cambia se vogliamo adattare gli esempi di questo articolo ad un linguaggio staticamente tipato, come Java o C#? L'unica vera differenza è nella classe base. Ad esempio, ecco una versione Java della classe Flight:

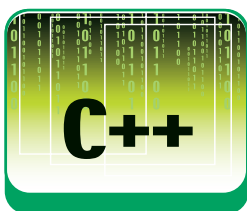
```
public abstract class Flight {

  private int free_seats;

  public Flight(int seats) {
    free_seats = seats;
  }

  protected void assignSeat(Customer c) {
    c.assign(free_seats);
    free_seats--;
  }

  public void book(Customer c) {
    if(free_seats == 0)
      throw new RuntimeException("Overbooking!");
    assignSeat(c);
    grantServices(c);
  }

  protected abstract void grantServices(Customer c);
}
```

Abbiamo dovuto dichiarare la classe abstract, e abbiamo dovuto scrivere una dichiarazione vuota del metodo grantServices(). In linguaggi dinamici come Python, Ruby o Perl, possiamo semplicemente omettere il metodo e considerare astratta la classe senza usare nessuna particolare parola chiave. Nessun compilatore si lamenterà, anche perché in questi linguaggi il compilatore non c'è. Il metodo assignSeat() ha un'implementazione nella classe base, quindi non cambia nulla tra la sua versione "statica" e quella "dinamica".
Un framework famoso che utilizza i Template Method è la libreria JUnit di Java, usata da moltissimi programmatori per scrivere i test unitari (ne esiste un equivalente per il mondo .Net, chiamato ovviamente Nunit). In questo framework tutti i test ereditano da una classe base TestCase, che contiene tra gli altri questo metodo:

```
public void runBare() throws Throwable {
  Throwable exception= null;
  setUp();
  try {
    runTest();
  } catch (Throwable running) {
    exception= running;
  }
  finally {
    try {
      tearDown();
    } catch (Throwable tearingDown) {
      if (exception == null) exception= tearingDown;
    }
  }
  if (exception != null) throw exception;
}
```

I metodi setUp(), tearDown() e runTest() sono anch'essi definiti da TestCase, ma gli ultimi due sono vuoti. Per scrivere un test, potete scrivere una sottoclasse di TestCase che contiene un metodo il cui nome inizia per test. Questo metodo viene trovato ed eseguito via reflection dal metodo runTest(). Potete anche specificare le operazioni di inizializzazione e chiusura del test ridefinendo i metodi setUp() e tearDown(), che nell'implementazione standard sono vuoti. Il metodo runBare(), che si occupa solo di decidere l'ordine delle operazioni e di gestire le eccezioni, è un caso da manuale di Template Method.

PROGRAMMIAMO UN LEAK-DETECTOR

CONTINUA LA SERIE SUL MEMORY MANAGEMENT IN C++. IN QUESTA PUNTATA VEDREMO COME SOVRACCARICARE IL COMPORTAMENTO STANDARD DEGLI OPERATORI NEW E DELETE PER REALIZZARE UN LEAK-DETECTOR SEMPLICE MA FUNZIONALE.

Se programmate in C++, vi trovate in una posizione invidiabile e critica nello stesso tempo. La parte invidiabile è che avete fra le mani un linguaggio unico, grazie al quale potete utilizzare strutture di altissimo livello, così come, quando occorre, potete dedicarvi ai più piccoli dettagli di basso livello. La parte critica, in tutto questo, è che se decidete di sfruttare tanta potenza siete tenuti a conoscere a fondo i meccanismi del linguaggio.
Questa serie, fin qui, ha mostrato il lato "di alto livello" dell'allocazione dinamica della memoria, illustrando nel dettaglio come funziona l'idioma RAII e il passando in rassegna di diversi smart pointer. Mettendo in pratica le teniche esposte fin qui, potete già programmare in maniera robusta e sicura. Ma quanto conoscete di ciò che avviene dietro le scene? Provate a rispondere a queste domande:

- Che differenza c'è tra l'"operator new" e il "new operator"?
- Come sono definiti, e cosa fanno?
- È possibile ridefinirne il comportamento?
- Cosa succede se operator new lancia un'eccezione?
- Che cos'è il "placement new"? E quanti ne esistono?
- Che cos'è, e com'è definito un allocatore STL?

Se avete risposto con un sonoro: "boh?" a uno o più di questi interrogativi, i vostri programmi non smetteranno certo di funzionare di colpo, ma è indubbio che vi state perdendo alcuni dettagli fondamentali del linguaggio che usate. In questa puntata risponderò a queste (e altre!) domande, presentandovi un progetto pratico: un vero e proprio **leak detector** – ovverosia un meccanismo che, una volta referenziato dal vostro programma, vi permetterà di rilevare se durante l'esecuzione si sono verificati dei memory leak, e vi aiuterà nel debug facendovi vedere dove avete allocato i dati, quanti byte sono andati persi, e addirittura "salvando" il vostro programma da pericolosi crash, in molti casi di double free.
Il risultato delle nostre fatiche non potrà certo rivaleggiare con i mastodonti più blasonati, ma vanterà alcuni grandi pregi: sarà leggero, portabile, potrà agganciarsi facilmente a programmi già esistenti, e soprattutto ci permetterà di scoprire come funzionano molti aspetti nascosti del C++.

OPERATOR NEW O NEW OPERATOR?

Chiariamo subito una grande ambiguità semantica del C++. Quando state allocando dinamicamente della memoria con istruzioni come:

```
string* nome = new string("Giabim");
```

state invocando il **new operator**. Questo è un componente del linguaggio il cui comportamento non è in alcun modo modificabile, e che si occupa di svolgere alcuni compiti fondamentali, in sequenza:

a) Calcolare quanta memoria serve per il dato.
b) Richiamare operator new per allocare la memoria.
c) Invocare il costruttore del tipo indicato (es: string("Giabim")) all'indirizzo ottenuto da operator new.

Qui salta fuori l'ambiguità di cui parlavo: ricordatevi di non confondere il **new operator** con l'**operator new**: sono due cose diverse, tanto è vero che l'uno richiama l'altro.
Operator new si occupa di allocare memoria nel free store e di restituire l'indirizzo ottenuto dal new operator.
Sebbene lo standard non imponga nessuna particolare implementazione di operator new,

un sistema piuttosto usato (nonché quello che seguiremo) consiste nel richiamare internamente la funzione di libreria **C malloc**. Un'implementazione minima è quella che segue:

```
void* operator new(size_t size) throw(std::bad_alloc)
                                                    {
        cout << "Ciao! Sto allocando "<< size << "
                                    byte!" << endl;
        return malloc(size);
}
```

Operator new è una funzione ridefinibile, quindi potete provare a scrivere quest'implementazione prima del main, e vedere che effettivamente, nel caso di una chiamata a new come quella all'inizio di questo paragrafo, il nostro operator new si comporta proprio come ci si aspetterebbe.

OPERATOR NEW E LE ECCEZIONI

Dunque operator new è davvero una funzione così banale? Non proprio: l'implementazione che abbiamo scritto sarà anche simpatica, ma è un po' troppo ottimista. Che succede se la size passata è 0? Che succede se malloc non riesce a trovare abbastanza spazio? Ecco una nuova implementazione che risponde a queste domande:

```
void* operator new(size_t size) throw(bad_alloc) {
        if (size == 0)
                size = 1;
        void* indirizzo = malloc(size);
        if (indirizzo != 0)
                return indirizzo;
        else
                throw bad_alloc();
}
```

Se la size passata è 0, operator new la incrementa, portandola a 1 (l'unità minima di allocazione); se malloc fallisce, operator new si limita a lanciare un'eccezione di tipo bad_alloc. In realtà, il giro dei "veri" operator new in questo caso dovrebbe comporsi di un ulteriore passaggio preliminare: l'invocazione della funzione ausiliaria new_handler, che può tentare di liberare spazio su disco. Se **new_handler** riesce nell'impresa, operator new può riprovare ad allocare il dato.
Se fallisce, sarà lo stesso new_handler a lanciare il bad_alloc. Poiché new_handler può essere scritto in maniera incrementale (ovvero: liberare un po' di spazio alla volta), è buona norma richiamarlo all'interno di un ciclo infinito, così:

```
//implementazione professionale di operator new
void* operator new(size_t size) throw(bad_alloc) {
        if (size == 0) size = 1;
        while(true) {
                void* indirizzo = malloc(size);
                if (indirizzo != 0)
                        return indirizzo;
                else {
                        //ottiene un puntatore
                          al new_handler globale
                        new_handler handler =
                            set_new_handler(0);
set_new_handler(handler);
                        //se è stato definito un
                          new_handler lo invoca
                        if (handler)
                                (*handler)();
                        else
                                throw
                                 bad_alloc();
                }
        }
}
```

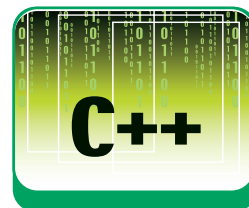
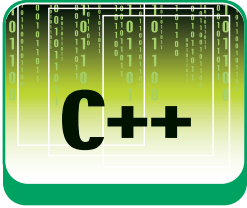
D'ora in poi, quando mostrerò il comportamento di un operator new, darò per scontata questa architettura sottostante.

PLACEMENT NEW

Non tutti lo sanno, ma il new operator può anche essere richiamato passandogli degli argomenti. Le versioni parametriche dei corrispondenti operator new prendono il nome di placement new. Eccone un esempio notevole:

```
void* operator new(size_t size, void* indirizzo)
                                 throw(bad_alloc) {
        return indirizzo;
}
```

Quello definito qui sopra, in particolare, è un placement new al quale viene passato un indirizzo, perché la memoria è già stata allocata. Poiché il suo lavoro è già stato svolto, questo si limita a... restituire l'indirizzo passato.
È una funzione tanto ricorrente che questo viene considerato il Placement New per antonomasia (e lo useremo anche noi, fra poco).
Ma è possibile definire placement new con un qualsiasi numero di argomenti. Noi, ad esem-

pio, sfrutteremo questa possibilità per memorizzare alcune utili informazioni, e cioè il nome del file e il numero di riga in cui è stato allocato il dato.

```
void* operator new(size_t size, const string&
                    fileName, int line) throw(bad_alloc);
        cout << "Ciao! In "<< fileName << ", alla
riga " << line << " hai allocato " << size "byte!" <<
                                                endl;
        return malloc(size);
}
```

Il programmatore che vorrà usufruire di questa comodità potrà richiamare new in questo modo:

```
string* nome = new("main.cpp", 1) string("Giabim")
```

Ovviamente, continueremo anche a supportare la versione "normale" di operator new, richiamando dal suo interno la versione estesa con argomenti come "[file non specificato]": non possiamo permetterci certo che tutto il codice non specificamente scritto per il leak detector venga gettato alle ortiche!

SITOGRAFIA

MyAllocator
Alla pagina http://www.josuttis.com/cppcode/allocator.html potete ammirare una splendida illustrazione di Josuttis (vedi Bibliografia) su come si scrive un allocatore STL completo.

Stones of nwva
http://sourceforge.net/projects/nvwa/
Fra le classi di questa piccola suite c'è un debug_new che è al contempo minimalista ed efficiente, ed è stato di una certa ispirazione per questo progetto.

Leak detector commerciali
Ci sono molti leak detector commerciali, sul mercato. La maggior parte di essi fa parte di un'applicazione più grande, come un debugger o un profiler. Operando in un processo separato, questi programmi possono analizzare una o più applicazioni dall'esterno, ed evidenziare le anomalie che avvengono in memoria durante l'esecuzione, oltre a varie statistiche. Ecco tre esemplari notevoli:

Rational Purify Plus
http://www-306.ibm.com/software/awdtools/purifyplus/
Conosciuto familiarmente come "purify", è uno dei codeProfiler/memoryDebugger tanto noti e usati da essere considerati uno standard de-facto, sotto UNIX, Windows e .NET.

GlowCode
http://www.glowcode.com/
Se siete sotto Windows, potete considerare GlowCode come una valida alternativa a Purify.

Intel VTune Performance Analyzer
www.intel.com/vtune
Se le vostre architetture di riferimento sono Intel e siete decisi ad individuare i colli di bottiglia e altre statistiche d'esecuzione sul vostro programma, questo strumento è semplicemente impareggiabile.

OPERATOR DELETE E PLACEMENT DELETE

Finora ci siamo occupati dell'allocazione di memoria, ma anche la deallocazione è una parte fondamentale. Fortunatamente il discorso a riguardo è quasi completamente speculare a quello fatto finora. Esiste, infatti, un **delete operator** il cui comportamento non è modificabile, e che si occupa di:

a) Richiamare il distruttore dell'elemento, e
b) Deallocare il blocco di memoria passato

Per soddisfare quest'ultimo punto viene richiamato l'**operator delete** (ormai comincia addirittura a suonarvi logico, vero?), che è sovraccaricabile, ed è definito così:

```
void operator delete(void* indirizzo) throw();
```

Niente eccezioni particolari, e nessun valore da restituire. Per deallocare il blocco utilizzeremo la funzione di libreria **C free**, come mostra quest'implementazione querula e minimalista:

```
void operator delete(void* indirizzo) throw() {
        if (indirizzo == 0)
                return;
        cout << "Addio! Sto dellocando i byte in
                        "<< indirizzo << endl;
        free(size);
}
```

L'unico discorso un po' differente e apparentemente contraddittorio riguarda il placement delete. Non esiste una sintassi specifica per richiamare il delete operator passandogli degli argomenti, in questo modo:

```
//questo codice è scorretto
delete(nome, "main.cpp", 2);
```

in ogni caso verrà sempre richiamato l'operator delete di base. Tutto questo potrebbe portarvi a credere (logicamente) che non esistano placement delete, ma vi sbagliereste. Infatti, potreste essere interessati a richiamare il solo operator delete (e non il delete operator!), così:

```
operator delete(nome, "main.cpp", 2);
```

Ma non vi esorto a farlo: dovreste svolgere anche il resto del lavoro del delete operator, mettendovi a richiamare esplicitamente il distruttore dell'oggetto (o i distruttori, se state deallocando un array), e non è una cosa troppo piacevole. La vera ragione per la quale si usano i placement delete è: per evitare memory leak. C'è una situazione critica, infat-

ti, che è potenzialmente a rischio:

a) Viene richiamata un'istruzione come: new Oggetto();
b) L'allocazione dell'operator new avviene con successo.
c) Il costruttore di Oggetto() lancia un'eccezione
d) Il programma cattura l'eccezione e continua
e) La memoria allocata non sarà mai rilasciata!

Per questo, se il costruttore lancia un'eccezione, il new operator la cattura e richiama automaticamente operator delete e distruttore dell'oggetto. Se si è usato un placement new per l'allocazione **viene cercato il placement delete corrispondente**. Se questo placement delete non è stato previsto, operator new **non fa nulla**. Per evitare memory leak, quindi, scriveremo anche la versione placement delete:

```
void operator delete(void* indirizzo, const string&
                         fileName, int line) throw();
        cout << "Ciao! In "<< fileName << ", alla
riga " << line << " hai allocato " << size "byte!" <<
                                                endl;
        return malloc(size);
}
```

FUNZIONAMENTO DEL LEAK DETECTOR

Ora ne sappiamo abbastanza per cominciare a stendere lo schema del nostro leak detector. Blaterare sullo schermo circa l'allocazione e la deallocazione della memoria non ha alcuna utilità pratica! Noi, invece, creeremo una struttura dati adatta a memorizzare queste informazioni:

```
struct MemoryRecord {
        string fileName;
        int line;
        size_t size;
        MemoryRecord(size_t size_ = 0, string
      fileName_ = "[Non specificato]", int line_ = -1) :
                size(size_), fileName(fileName_),
                                    line(line_) {};
};
```

Il piano d'azione è questo:
a) Dichiariamo un contenitore globale di MemoryRecord (chiamiamolo heapObjects)
b) Quando viene richiamato l'operator new, aggiungiamo un nuovo record a heapObjects
c) Quando viene richiamato l'operator delete, togliamo l'oggetto dalla lista
d) Alla fine del programma, heapObjects dovrebbe essere vuoto. Se così non è, stampiamo un messaggio di memory leak per ogni elemento.

Sembra tutto molto bello e lineare. C'è solo un ultimo, piccolo problema da risolvere: che contenitore usiamo? Normalmente non avremmo dubbi: un map<void*, MemoryRecord> sembra fatto apposta. Ma, come dicevo, c'è un problema. Il nostro programma dovrà fare qualcosa del genere:

```
void* operator new(size_t size, const string&
                   fileName, int line) throw(bad_alloc);
void* indirizzo = malloc(size);
heapObjects[indirizzo] = MemoryRecord(size,
                                      fileName, line);
        return indirizzo;
}
```

Forse avete già capito dove sta il problema: che istruzione pensate userà il contenitore map per l'inserimento del record? Il new operator, bravi! E così cadiamo in un ciclo infinito nel quale per ogni allocazione di un nuovo record… si alloca un altro record. Un'ottima maniera per esaurire le risorse disponibili. Allo stesso modo non è possibile usare direttamente le stringhe all'interno di operator new, dal momento che la loro implementazione potrebbe allocare memoria dinamicamente. Questo problema ha almeno tre soluzioni:

a) Usiamo dei contenitori fatto da noi (reinventiamo la ruota).
b) Usiamo strutture e contenitori C di terze parti (aggiungiamo una dipendenza).
c) Usiamo i contenitori standard senza fargli usare operator new.

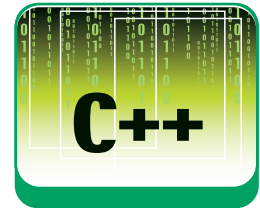
L'ultimo punto sembra promettente. Ma come fare? Benvenuti nel mondo degli allocatori STL.

MALLOCATOR E GLI ALLOCATORI STL

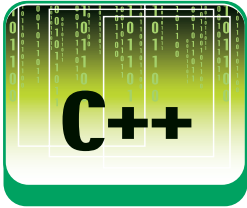
I contenitori della libreria standard non usano new e delete direttamente, ma fanno riferimento a una classe detta **allocatore**, che astrae il concetto dell'allocazione in memoria. Un allocatore ha sostanzialmente due compiti:

a) Fornire ai contenitori una serie di typedef (francamente piuttosto ridondanti), come:

```
typedef T              value_type;
typedef T*             pointer;
typedef const T* const_pointer;
e così via per reference, const_reference, etc…
```

b) Fornire una serie di metodi, fra i quali gli

BUON VECCHIO NEW

Esiste anche una "vecchia" implementazione di new, che non lancia eccezioni. Questa è richiamabile attraverso un particolare placement new, con argomento std::nothrow. Ad esempio:

```
string =
new(std::nothrow)
string("Giabim")
```

essenziali:

```
//alloca un oggetto (senza costruirlo)
pointer allocate(size_type n);
//dealloca un oggetto (senza distruggerlo)
void deallocate(pointer p, size_type n);
//costruisce un oggetto in un'area già allocata
void construct(pointer p, const_reference v);
//distrugge un oggetto senza deallocare l'area di
                                                    memoria
void destroy(pointer p);
```

I contenitori standard usano per default l'oggetto allocator, che richiama new e delete, ma si può anche specificare di usarne un altro. E infatti noi creeremo un allocatore che non passerà per gli operator new, e che chiameremo con un estenuante sforzo di fantasia: Mallocator. Realizzarlo da zero non è un compito difficile, ma è tedioso, e soprattutto lungo (l'articolo finirebbe qui), quindi useremo una scorciatoia: erediteremo da allocator, e ne ridefiniremo solo i typedef e le funzioni chiave. Se avete dei bambini, dite loro di non farlo a casa: non è una pratica sicura sovraccaricare metodi nonvirtuali - su classi, peraltro, con un distruttore nonvirtuale. Ma allocator è una classe banale, e la cosa non dà alcun problema.

```
template<typename T> struct Mallocator :
                                        allocator<T>
{
typedef T           value_type;
        typedef T*          pointer;
        //... eccetera ...
        template <class U> struct rebind {
                typedef Mallocator<U> other;
        };
        Mallocator() {};
        template <class U> Mallocator(const
                        Mallocator<U>& b) {};
        pointer allocate(size_type n) {
                T* address = (T*)
                        malloc(sizeof(T) * n);
                return address;
        }
        void deallocate(pointer p, size_type n) {
                free(p);
        }

        void construct(pointer p, const_reference v)
                                                    {
                new(p) T(v);
        }
        void destroy(pointer p) {
                p->~T();
        }
};
template<class T> bool operator ==(const
```

```
    Mallocator<T>& a1, const Mallocator<T>& a2) {
        return true;
}
template<class T> bool operator !=(const
    Mallocator<T>& a1, const Mallocator<T>& a2) {
        return false;
}
```

A parte l'ingegnoso giochino del rebind per permettere il costruttore per copia, che probabilmente vi stupirà se non avete una grande esperienza con la programmazione per template, tutto il resto è piuttosto lineare. La cosa che potrebbe sorprendervi è il ritrovarvi ancora fra i piedi una chiamata a new, nel metodo construct. Ma non si era deciso di abolirlo? Guardatelo bene: è Il Placement New (quello per antonomasia), che serve appunto a costruire un oggetto in una zona di memoria già allocata (infatti l'operator new corrispondente non fa praticamente nulla).
Ora possiamo creare qualsiasi tipo di contenitore standard, senza fargli usare new. Nel nostro caso, il tipo di una map<void, MemoryRecord> è:

```
typedef std::map<
        void*,
        MemoryRecord,
        less<void*>,
        Mallocator< pair<const void*,
                        MemoryRecord> >
> MallocMap;
```

E una string diventa:

```
typedef std::basic_string<
        char,
        char_traits<char>,
        Mallocator<char>
> MallocString;
```

Grazie a questi typedef, ora possiamo archiviare l'intera faccenda, semplicemente usando i contenitori MallocMap e MallocString.

ALLOCATEMEMORY E DEALLOCATEMEMORY

Abbiamo un piano d'azione, e tutti gli elementi per portarlo a termine. Cercherò di illustrare i punti salienti dell'implementazione del progetto, rimandandovi al file allegato per gli aspetti più banali. Cominciamo a delinearne la struttura, partendo dagli operator new.

```
void* operator new(size_t) throw(bad_alloc);
void* operator new(size_t, const MallocString&, int)
                                    throw(bad_alloc);
```

BIBLIOGRAFIA

Scott Meyers Effective C++ (l'intero capitolo 8 è una miniera d'informazioni sull'argomento) More Effective C++ (Item 8 e 9) Effective STL (l'item 10 spiega chiaramente come NON si usano gli allocatori STL). Nicolai M. Josuttis The C++ Standard Library - A Tutorial and Reference

```
void* operator new[](size_t size) throw(bad_alloc);
void* operator new[](size_t size, const
                MallocString&, int) throw(bad_alloc);
```

Come vedete, ho sovraccaricato anche gli operatori new[], che sono concettualmente identici a quelli new. Ovviamente, occorrerà sovraccaricare anche i corrispettivi operatori delete[].

```
void operator delete(void* address) throw();
void operator delete(void* address, const
            MallocString& fileName, int line) throw();
void operator delete[](void* address) throw();
void operator delete[](void* address, const
            MallocString& fileName, int line) throw();
```

poiché tutte queste funzioni fanno la stessa cosa, si limitano a chiamare le funzioni LeakDetector::AllocateMemory, e LeakDetector::DeallocateMemory, con i giusti argomenti.

```
namespace LeakDetector {
        void* AllocateMemory(size_t size, const
                MemoryRecord& info, bool isVector);
        void DeallocateMemory(size_t size, const
                MemoryRecord& info, bool isVector);
}
```

Entrambe le funzioni, così come tutto il resto del leak detector, fanno parte del namespace LeakDetector – in modo da evitare confusione e conflitti con nomi già esistenti.
Il comportamento di AllocateMemory è già stato spiegato nei paragrafi precedenti, mentre DeallocateMemory ha bisogno di qualche spiegazione:

```
void DeallocateMemory(void* address, const
                MemoryRecord& info, bool isVector)
{
        if (heapObjects.find(address) ==
                        heapObjects.end()) {
                cerr << "LEAK DETECTOR:
    Rilevato un probabile double free su " << address;
                cerr << "LEAK DETECTOR:
                        Deallocazione ignorata.";
                return;
        }
        free(address);
        heapObjects.erase(address);
}
```

Il comportamento dell'ultima parte è intuitivo: si libera la memoria all'indirizzo passato, e si tolgono da heapObjects le informazioni relative. In condizioni normali, questo è perfetto; in condizioni patologiche, invece, no. Consideriamo questo caso:

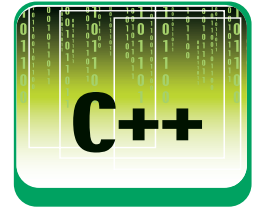
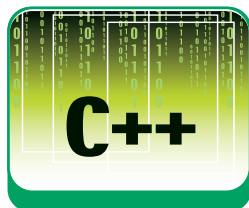
```
int* numero = new int();
delete numero;
delete numero;
```

La prima chiamata a delete dealloca la memoria e rimuove l'indirizzo dal contenitore, ma alla seconda, la chiamata ad heapObjects.erase() fallirebbe. E soprattutto, l'istruzione free() provocherebbe un double free. Così si capisce il senso dei controlli iniziali: se l'oggetto non è nella lista, vuol dire che non è stato allocato, o che è stato già deallocato. In questo caso, il leak detector stampa un messaggio di avvertimento, e evita la deallocazione, per permettere al programma di proseguire.
Se avete compreso perfettamente il funzionamento del delete operator potete già immaginare che queste contromisure possono non essere sufficienti ad impedire il crash dell'applicazione. Il problema è la distruzione dell'oggetto, che avviene prima della chiamata all'operator delete. Il leak detector sarà quindi in grado di prevenire una situazione come quella mostrata qui sopra, ma nulla potrà contro il delete di oggetti dotati di un costruttore complesso, che faccia riferimento ai dati interni già distrutti della classe. Come string, ad esempio:

```
string* stringa = new string();
delete stringa;
delete stringa; //oggettoCorrotto->~string()? Crash!
```

CHECKLEAKS

Veniamo adesso al punto più importante dell'analizzatore di leak… analizzare i leak! La funzione *LeakDetector::CheckLeaks* sfoglia l'elenco dei record presenti nel contenitore, e stampa su cerr le informazioni relative.

```
void CheckLeaks() {
        for (MallocMap::iterator i =
    heapObjects.begin(); i != heapObjects.end(); ++i) {
                cerr << "LEAK DETECTOR: Rilevato
                                memory leak.\n"
                        << (string)i->second
                                << endl
                        << "Contenuto di " <<
                                i->first << ":\n";
                //esegue un dump della memoria
                unsigned char* address =
        reinterpret_cast<unsigned char*>(i->first);
                for (unsigned char* p=address; p
                < address + i->second.size; p++) {
                        cerr << hex <<
```

```
            static_cast<int>(*p) << " ";
        }
        cerr << endl << endl;
    }
}
```

Come vedete, la funzione si occupa anche di eseguire un dump del segmento di memoria interessato (per un esempio, guardate la Figura 1).
Uno dei punti critici della funzione, nonché dell'intero progetto, è stabilire come e quando questa debba essere richiamata. Un modo semplice sarebbe quello di dare quest'incombenza al programmatore finale:

```
int main() {
    //fa quel che deve fare
    LeakDetector::CheckLeaks();
    return 0;
}
```

Una simile idea è una base di partenza, ma è anche piuttosto ingenua. Non tanto perché l'utente potrebbe dimenticarsi di scrivere la chiamata (quello è un problema grave in istruzioni che vanno ripetute continuamente, mentre questa è una sola). L'idea ingenua è che il programma finisca quando finisce il main. Se avete seguito gli appuntamenti precedenti, avete già un'eccezione sotto al naso:

```
int main() {
    auto_ptr<string> nome(new("main.cpp", 2)
                    string("ciao"));
    LeakDetector::CheckLeaks();
    return 0;
}
```

In questo programma il detector segnalerà un leak che non esiste. Infatti al momento della chiamata a CheckLeaks(), la memoria allocata per "nome" non sarà ancora deallocata. Ma lo sarà dopo, all'uscita dal main, allo srotolamento dello stack. Questa è solo la punta dell'iceberg: pensate anche agli oggetti statici e alle costanti. Fuori dal main, insomma, c'è tutto un mondo potenzialmente pieno di memoria dinamica non ancora deallocata.
Per non creare problemi di compatibilità, nel file allegato ho adottato una soluzione non proprio ottimale, ma molto comprensibile: ho fatto sì che, alla prima esecuzione, AllocateMemory registrasse CheckLeaks in AtExit. Così:

```
static bool firstCall = true;
    if (firstCall) {
        atexit(LeakDetector::CheckLeaks);
        firstCall = false;
    }
```

Ci sono varie tecniche più complicate, interessanti e convenzionali per ovviare al problema (Singleton? Phoenix? Altro?), potete divertirvi a provarle sul codice e vedere pregi e svantaggi. Uno dei sistemi più indicati potrebbe essere quello normalmente usato per implementare cin, cout, e gli altri stream predefiniti, che soffrono dello stesso problema. L'idea è quella di creare un sistema di reference counting, che tenga traccia del numero di unità di traduzione ancora in vita. Quando il conto arriva a zero, tutto è stato deallocato.

```
namespace LeakDetector {
    struct Counter {
        static int count;
        Counter() {
            Count++;
        };
        ~Counter() {
            if (--count == 0)
                CheckLeaks();
        };
    };
}
namespace {
    LeakDetector::Counter counter;
}
```

CONCLUSIONI

Con la "scusa" di implementare il nostro leak detector, ci siamo addentrati piuttosto a fondo negli ingranaggi che regolano l'allocazione dinamica della memoria in C++. Il progetto, sebbene sia limitato dai suoi stessi fini didattici, è abbastanza sofisticato da essere realmente utile. Al prezzo di complicare un po' tutto, si potrebbero fare altre meraviglie. Ad esempio, non sarebbe difficile implementare un sistema di bounds checking, o iniettare delle informazioni direttamente nella memoria allocata (il problema principale, in entrambi i casi, è conoscere e preservare il corretto allineamento dei dati).
Lascio a voi, se vorrete, l'implementazione di tutte queste migliorie: questa serie deve andare avanti verso l'ultima puntata, in cui risaliremo all'alto livello, arrivando a raggiungere il Regno dei Garbage Collector. Non mancate!

Roberto Allegra

L'AUTORE

Per ogni richiesta/critica/suggerimento l'autore può (e deve!) essere contattato all'indirizzo
articoli@robertoallegra.it

MOBILE SEMPLICE CON JAVA E NETBEANS

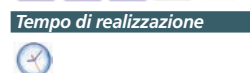
SE SIETE ABITUATI A QUALCHE TRILIONE DI RIGHE DI CODICE PER PROGRAMMARE APPLICAZIONI PER CELLULARI E PALMARI, RIMARRETE A BOCCA APERTA DI FRONTE ALLA PROGRAMMAZIONE RAD POSSIBILE CON I NUOVI STRUMENTI VISUAILI...

Le applicazioni per dispositivi mobili sono in continua evoluzione così come le tecnologie necessarie alla loro implementazione. In questo scenario la piattaforma Java 2 Micro Edition di Sun Microsystems occupa una posizione predominante. Tale successo, oltre ad essere determinato dalla enorme diffusione del linguaggio ideato da James Gosling, è principalmente attribuibile al fatto che la piattaforma in questione è distribuita *embedded* all'interno di una moltitudine di dispositivi, primi fra tutti i telefoni cellulari. A causa delle attuali limitazioni hardware, la stragrande maggioranza di questi device è equipaggiata con lo strato CLDC (Connected Limited Device Configuration) che mette a disposizione degli sviluppatori un set di librerie drasticamente ridotto e pertanto viene utilizzato per la realizzazione di applicazioni di piccole dimensioni. Grazie all'evoluzione dei dispositivi ed alla progressiva miniaturizzazione delle componenti hardware è possibile implementare applicazioni basate su CDC (Connected Device Configuration) che, come vedremo nel proseguo di questo articolo, offrono allo sviluppatore una API comparabile a quella presentata dalla piattaforma J2SE e, con l'aggiunta di packages opzionali, permettono l'utilizzo di librerie grafiche avanzate.

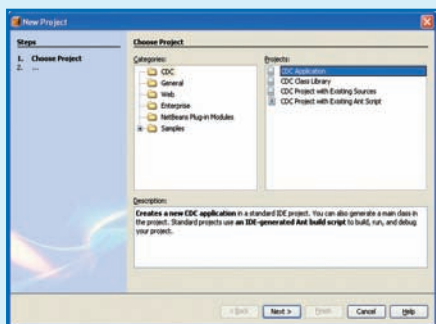
CDC IN DETTAGLIO

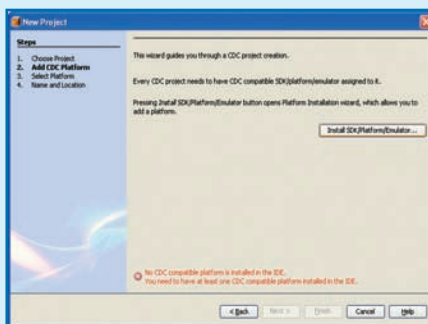
La versione 1.1 dello standard CDC risponde alle esigenze formalizzate con la Java Specification Request 218 ed è ottimizzata per dispositivi equipaggiati di microprocessori a 32 bit. Per il corretto funzionamento delle applicazioni costruite su questo strato, il dispositivo deve mettere a disposizione della Java Virtual Machine almeno 256K di RAM e 512K di ROM. Le categorie di device che possono ospitare questa configurazione sono molteplici; tra le più comuni si possono elencare: i moderni smartphone, i più evoluti PDA, i set-top box televisivi fino ad arrivare alle stampanti multifunzione ed i dispositivi embedded. Da un punto di vista di linguaggio e di librerie messe a disposizione dello sviluppatore, lo strato CDC può essere visto come un superset del CLDC ed una semplificazione della J2SE. Tra le più importanti componenti "ereditate" dalla versione standard si possono elencare: la reflection, JNI (Java Native Interface) ed il sistema di networking. Così come il MID Profile per configurazioni CLDC, sullo strato CDC si possono aggiungere profili aggiuntivi o packages opzionali in modo da aumentarne le potenzialità. Nelle successive sezioni introdurremo il neonato package AGUI (Advanced Graphics and User Interface – JSR 209) che consente la creazione di interfacce grafiche tramite componenti JFC/Swing e la tecnologia Java 2D.
Per meglio comprendere il funzionamento e le potenzialità del CDC, il "primo esercizio" che faremo sarà l'implementazione di una semplice applicazione per la consultazione delle riviste di ioProgrammo. Attraverso una combo-box sarà possibile scegliere il numero desiderato, la cui copertina verrà visualizzata al centro del display.

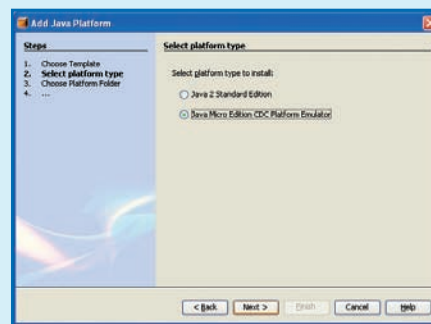
CREAZIONE DELLA GUI

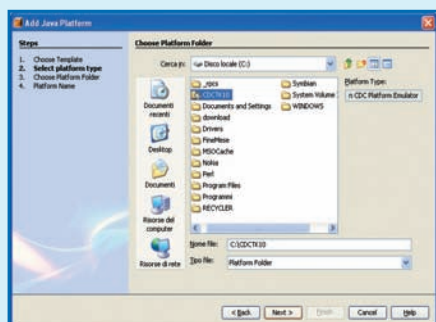
Il pacchetto comprendente il Sun Java Toolkit for CDC viene distribuito insieme ad un ambiente di sviluppo "ad hoc" per questo tipo di applicazioni. Questo IDE è basato su Netbeans ma ne comprende solo un piccolo subset di funzionalità. Pertanto, nel corso di questo articolo, per facilitare e velocizzare l'implementazione della nostra applicazione, si è scelto di utilizzare l'ultima versione di Netbeans disponibile, corredata con il relativo Mobility Pack for CDC. L'integrazione tra l'IDE ed il Toolkit è descritta nell'apposito riquadro.
Chi si fosse già cimentato nella creazione di inter-

NETBEANS E JAVA TOOLKIT FOR CDC IN SEI PASSI

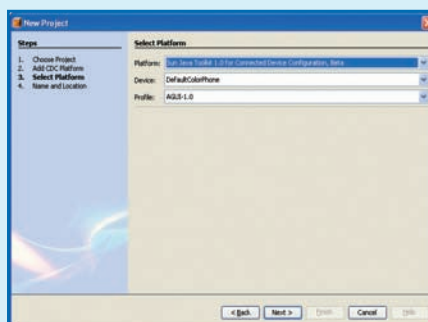
Per utilizzare il Sun Java Toolkit for CDC in Netbeans è necessario integrare la piattaforma comprendente l'emulatore all'interno dell'ambiente di sviluppo. Le istruzioni sono riportate nel seguente riquadro.

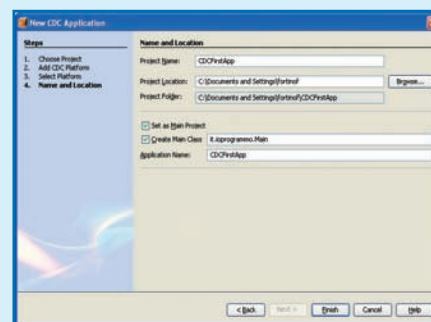
> UN NUOVO PROGETTO

1 Creiamo un nuovo progetto dal menu File -> New Project

> SCEGLI LA PIATTAFORMA

2 L'IDE ci aobbliga a scegliere la piattaforma CDC da utilizzare

> QUALE TIPO?

3 Specifichiamo di voler installare una piattaforma J2ME con CDC.

> IL SUN TOOLKIT

4 Selezioniamo la directory in cui abbiamo installato il Sun Toolkit for CDC

> IMPOSTAZIONI GENERALI

5 Possiamo tranquillamente lasciare le impostazioni di default

> I PACKAGE

6 Specifichiamo nome, package, directory etc. et voilà!

facce grafiche per applicazioni desktop con il Matisse GUI Builder di Netbeans, si sentirà immediatamente a suo agio: lo stesso strumento viene infatti riproposto per la creazione di applicazioni CDC. Attraverso questo tool, molto potente ed allo stesso tempo estremamente intuitivo, è possibile creare in maniera visuale, attraverso la tecnica del drag-and-drop, le GUI delle nostre applicazioni mobili senza soffermarsi sulle complessità insite all'interno delle componenti Swing. In **Figura 1** è illustrata la vista "Design" di Netbeans raffigurante la struttura della GUI dell'applicazione di esempio: la *JComboBox* e le due *JLabel* (una posta alla sinistra della combo-box, l'altra contenente l'immagine di copertina) sono state semplicemente trascinate dalla *Palette* posta alla destra dell'IDE. In background, il GUI Builder ha "tradotto" le nostre decisioni in componenti e funzionalità grafiche applicando il potente layout manager *GroupLayout* che facilita il posizionamento delle componenti stesse all'interno del contenitore grafico. Per chi volesse visualizzare il codice generato è possibile selezionare la vista "Source".

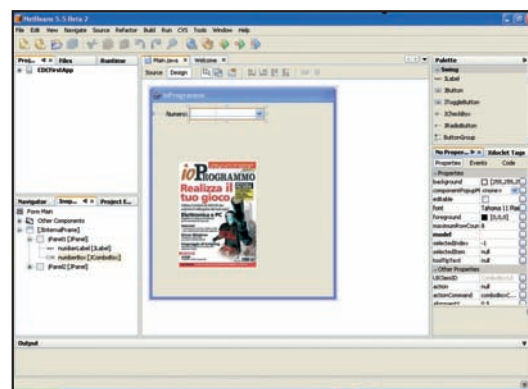



Fig. 1: ***Struttura grafica dell'applicazione di esempio creata con il Matisse GUI Builder di Netbeans***

INTEGRIAMO IL CODICE GENERATO

Una volta completato il disegno dell'interfaccia grafica dell'applicazione, non ci rimane che integrarla con la restante parte logica. Le funzionalità non ancora implementate sono due: l'inizializzazione dei valori presentati dalla combo-box e la visualizzazione delle copertina corrispondente al valore selezionato. In entrambi i casi il GUI Builder ci viene incontro guidandoci passo dopo passo. Per quanto riguarda il primo punto è sufficiente editare le proprietà avanzate dell'attributo model della combo-box, inserendo il seguente stralcio di codice:

```
String[] modelArray = new String[47];
for (int i = 0; i < modelArray.length; i++) {
    modelArray[i] = "" + (i+59);
}
this.numberBox.setModel(new
   javax.swing.DefaultComboBoxModel(modelArray));
```

Le copertine a nostra disposizione sono 47 e si riferiscono ai numeri di ioProgrammo che vanno dal 59 al 105. Pertanto è necessario creare un array, i cui riferimenti sono delle stringhe che rappresentano tali valori, ed utilizzarlo per inizializzare il modello dati della componente combo-box.
Per fare in modo che al cambio di valore corrisponda la visualizzazione della relativa immagine, è invece sufficiente aggiungere un handler all'evento itemStateChanged della combo-box, che, nel caso specifico, chiameremo changeRivista. L'ambiente di sviluppo, a questa nostra richiesta, registra automaticamente il listener sul componente e lo associa al metodo changeRivista, che viene inizialmente creato vuoto. È nostro compito completarlo con il seguente codice:

```
private void changeRivista(java.awt.event.ItemEvent
                                                evt) {
        // TODO add your handling code here:
        Object nRivista =
                    this.numberBox.getSelectedItem();
        this.coverImageLabel.setIcon(new
 javax.swing.ImageIcon(getClass().getResource("/res
                      /4-" + nRivista + "g.jpg")));
    }
```

che, individua l'item selezionato, recupera l'immagine associata e la sostituisce alla precedente. A questo punto, non ci rimane che eseguire l'applicazione all'interno dell'emulatore DefaulColorPhone , fornito con il Java Toolkit for CDC, come illustrato in Figura 2.

Fig. 2: ***La nostra applicazione all'interno dell'emulatore per piattaforma J2ME CDC.***

CONCLUSIONI

I dispositivi mobili sono in continua evoluzione. Questo progresso permette la creazione di applicazioni sempre più potenti e graficamente accattivanti. Per quanto riguarda la tecnologia J2ME questa evoluzione è rappresentata dal Connected Device Configuration e dai pacchetti aggiuntivi come Personal Profile o AGUI. Al momento, i dispositivi che supportano questi standard non sono molti ed hanno un costo di acquisto medio-alto. Come sempre succede con l'evolvere della tecnologia i costi si abbatteranno e questi prodotti diventeranno di uso comune. In questo articolo abbiamo visto come realizzare con facilità una semplice applicazione per dispositivi mobili costruita su questo strato applicativo attraverso l'utilizzo del Matisse GUI Builder di Netbeans. Sebbene CDC sia uno standard ancora giovane, è importante sottolineare che, con questa nuova tecnologia J2ME non focalizzerà più gli sviluppi su un ristretto, se non unico, target di device, ma aprirà le porte degli sviluppatori ad una vasta gamma di dispositivi di uso quotidiano.

Fabrizio Fortino

SOFTWARE SUL CD

JDK 1.6

LA NUOVA VERSIONE DEL COMPILATORE JAVA

Finalmente è arrivato. Ce ne parla Federico Paparoni nell'articolo di copertina di questo mese. Con le nuove estensioni per il Desktop e la maggiore integrazione con il sistema operativo d'origine la nuova versione di Java oltre che un'innovazione tecnologica fa segnare un pesante cambiamento di rotta di tipo filosofico, staremo a vedere quanto questa nuova direzione inciderà nel mercato dello sviluppo multipiattaforma

Directory:/jdk-6-windows-i586.exe

APACHE 2.2.3

IL WEB SERVER CHE HA FATTO INTERNET

Una colonna portante della rete. Apache è realmente il server più diffuso su Internet, e non si contano i moduli disponibili per estenderne le funzionalità. In questo numero di ioProgrammo lo utilizziamo in congiunzione a SVN che consente di tenere traccia delle modifiche apportate allo sviluppo di un'applicazione. Apache supporta pienamente linguaggi come PHP, Perl, Python e Ruby, ma anche tanti altri.

Directory: /apache_2.2.3-win32-x86-no_ssl.zip

DOTNETNUKE 4.3.7 STARTER KIT

IL PORTALE "PRECOTTO" TARGATO .NET

Un CMS completo e dotato di funzionalità realmente avanzate. In questo numero presentiamo uno "Starter Kit" ovvero un template per l'ambiente Visual Studio, che consente di installare il prodotto ma anche di estenderlo per coloro che avessero bisogno di funzionalità particolari.

Directory: /DotNetNuke_4.3.7_Source.zip

MEDIAWIKI 1.8.2

IL LEADER PER LO SVILUPPO DI APPLICAZIONI DI WIKI

Chi non conosce wikipedia? l'enciclopedia libera? Il funzionamento è semplice. Si punta il browser verso un link non esistente e l'applicazione crea per voi un contenuto che ha per titolo il nome del link che avete puntato. I contenuti vengono messi in relazione fra loro tramite un linguaggio interno. Questo è un wiki! Mediawiki è il software che sta alla base di wikipedia e che anche voi potete utilizzare per creare ad esempio della documentazione relativa all'applicazione che avete appena sviluppato.

Directory: /mediawiki-1.8.2.tar

MRTG 2.15

MISURAZIONI DI RETE PERFETTE

Volete sapere quanto consuma in termini di banda il vostro WebServer? Avete necessità di conoscere quanta banda viene consumata in termini di accessi FTP? Mrtg è un frontend verso il servizio SMTP . Consente di misurare con estrema precisione tutti i parametri che caratterizzano il vostro sistema sia esso un sistema Window che Linux. Requisiti fondamentali per utilizzare a fondo questo frontend sono un'installazione di Perl funzionante sulla propria macchina e che il servizio Snmp sia attivo.

Directory: /mrtg-2.15.0

PHPBB 3.0.B4

IL FORUM PER ECCELLENZA

Scritto in PHP con l'ausilio di MySQL è stata probabilmente la prima applicazione OpenSource a proporre un modello per lo sviluppo di forum sul Web. Attualmente è probabilmente il prodotto che espone il maggior numero di funzionalità, più facilmente estendibile e più ricco di moduli. In alcune versioni sono stati riscontrati problemi di sicurezza, tuttavia prontamente risolti. Se avete bisogno di installare un forum o più semplicemente volete studiare alcune tecniche per scrivere codice elegante e funzionale in PHP, probabilmente PHPbb è il software che fa per voi

Directory: /phpBB-3.0.B4.zip

ZEND GOOGLE DATA 0.6.0

IL FRAMEWORK PER SCRIVERE SUL WEB

Google recentemente sta ampliando la sua gamma di prodotti proponendo agli sviluppatori una serie di API che consentono di interfacciare i propri programmi con i servizi messi a disposizio-

ne dal motore di ricerca. Fra le API più recenti compare questa Google Data, ovvero un protocollo che consente di scambiare i dati secondo regole ben precise sul Web. Zend ha immediatamente colto la palla al balzo proponendo questo framework che rende più immediato l'interfacciamento di questo protocollo con le applicazioni PHP. Non mancheremo di approfondire questo argomento nei prossimi numeri di ioProgrammo

Directory: /gdata.zip

ANT 1.6.5

IL MAKE DI JAVA

Tutti conosco l'utility make. Il suo funzionamento è facile. Legge un file di configurazione e compila e installa un prodotto sulla base delle opzioni in esso contenute. Ant è un tool che espleta una funzione molto simile a quella di Make, ma più orientata al codice Java. Rispetto a make utilizza alcuni concetti avanzati, come la lettura delle opzioni di configurazione direttamente da un file XML. Inoltre mentre make prende parametri da una linea di comando, Ant può essere esteso direttamente da classi Java

Directory: /apache-ant-1.6.5-bin.zip

DEVCPP 4.9.2

IL PIÙ AMATO DAI PROGRAMMATORI C++

Da qualche tempo non faceva capolino fra le nostre pagine questo interessantissimo IDE per programmatori C++. Sicuramente si tratta di uno degli ambienti più amati dai programmatori di questo linguaggio. Dalla sua parte un'incredibile leggerezza che lo rende utilizzabile su macchine anche non eccezionalmente performanti, e poi tutta una serie di features che vanno dal code complexion alla sintax highlightinhg, ma non solo. Si tratta di un software completo che esteso con plugin diventa anche un editor RAD con interfaccia visuale.

Directory: /devcpp-4.9.9.2_setup.exe

IRONPYTHON 1.0.1

UN'IMPLEMENTAZIONE DI PYTHON IN TECNOLOGIA .NET

In molti conoscono Python. Si tratta di un linguaggio agile, dinamico, elegante, ad oggetti e multipiattaforma. Per le sue caratteristiche di estrema flessibilità, per la sua facilità nella curva di apprendimento, per la sua leggerezza si tratta di un linguaggio estremamente diffuso su piattaforma Linux dove viene utilizzato per automatizzare gran parte delle impostazioni di sistema. Anche sul Web Python ha trovato una sua collocazione ben precisa, pare infatti che sia il linguaggio di scripting su cui si basa buona parte di Google. In ambiente Windows si sta diffondendo a macchia d'olio, tanto che oltre ad una versione standalone dell'interprete ne è appena nata anche una versione per .NET, appunto IronPython. Le caratteristiche sono identiche a quelle classiche di Python, ma il codice prima di essere eseguito viene convertito nel MIL di .NET con le conseguenti ottimizzazioni

Directory: /IronPython-1.0.1-Bin.zip

J2ME POLISH 1.2.4

IL COSTRUTTORE DI GUI PER DEVICE MOBILI

Si tratta di un software il cui scopo è supportare il programmatore nella creazione di interfacce destinate a dispositivi portatili quali ad esempio cellulari o palmari. Ovviamente la base su cui si fonda è J2ME ormai onnipresente in qualsiasi applicazione per dispositivi del genere, tuttavia invece la definizione dell'interfaccia si basa su semplici file HTML con la stessa logica che usano spesso i template che separano il codice dal layout di un'applicazione. Interessante è anche il fatto che J2ME sia distribuito sotto licenza GPL

Directory: /j2mepolish-1.2.4.jar

JAMES 2.3.0

IL MAIL SERVER DI APACHE

Completamente scritto in Java è dotato di tutte le caratteristiche essenziali di un ottimo Web Server. Gestisce SMTP e POP3, filtri, blacklist ed è sufficientemente performante da poter essere utilizzato in fase di produzione. Quel che più conta è estendibile perciò facilmente controllabile e personalizzabile da chi ne fa uso

Directory: /james-2.3.0

MANTIS 1.0.6

OTTIMA SOLUZIONE PER LA GESTIONE DEI BUG

Potete star certi che qualunque sia il numero di test effettuati prima di mettere un software in commercio, nel momento in cui avrà successo e sarà usato da un pubblico vasto verrete sommersi da indicazioni di Bug o richieste di nuove features Mantis è una Web Application che consente agli utenti registrati di indicare un bug reperito nella vostra applicazione di modo che voi possiate avere un comodo registro dei bug, e utilizzarlo per risolverlo ed elaborare le patch in modo adeguato

Directory: /mantis-1.0.6

NANT 0.85

L'ASSEMBLATORE DI PROGETTI

Nant è un tool che consente di costruire un processo automatico per la generazione di applicazioni. Da utilizzare quando un prodotto è suddiviso in molte dll o sottoprodotti gestiti da reparti separati, nant crea una specie di processo batch che raccoglie i file dalle directory dei singoli sviluppatori, li compila e li predispone per la costruzione di un'applicazione completa. Si tratta dell'omologo di Ant per Java e risulta molto comodo in progetti di grandi dimensioni, oppure quando si fa riferimento a una serie di DLL che interagiscono fra loro. E' uno dei tool più utilizzati in tecnologia .NET anche se recentemente alcune nuove features di Visual Studio 2.0 possono essere utilizzate in congiunzione a Nant per ottenere installazioni molto complete. Stiamo parlando ovviamente di ClickOnce

Directory: /nant-0.85-bin.zip

NOTEPAD ++ 3.9

L'EDITOR LEGGERO PER GLI SVILUPPATORI

Notepad++ è un editor di codice per programmatori e un potente editor di testi per utenti comuni: leggero, versatile e altamente configurabile che integra numerosi algoritmi di codifica crittografica e alcuni utili tool. Il programma è realizzato interamente in C++, mentre i linguaggi supportati sono: C/C++, Java, HTML, XML, PHP, JavaScript, Visual Basic, Perl, Python, CSS e tanti altri. Notepad++ è distribuito con licenza GNU GPL e alcune sue parti derivano dal progetto Scintilla.

Directory: /npp.3.9.Installer.exe

OPENLASZLO 3.3.3

QUASI COME FLEX MA OPENSOURCE

L'idea è semplice. C'è un file XML, l'utente richiama questo file per mezzo del browser. Il compilatore sul server lo compila on the fly e restituisce all'utente una pagina flash. La tecnica è interessante perché consente di creare applicazioni WEB 2.0 senza utilizzare AJAX e producendo sempre output adeguati e compatibili con ogni browser.

Directory: /openlaszlo-3.3.3-windows-dev-install.exe

PHPECLIPSE 1.1.8

PER USARE ECLIPSE CON PHP

Molti di voi conosceranno Eclipse, si tratta dell'ambiente "Tuttofare" con cui ormai è possibile programmare qualunque tipo di linguaggio o quasi. Pur essendo fortemente orientato verso java, Eclipse è esensibile tramite plugin per essere utilizzato per diversi scopi. Il plugin che qui vi presentiamo estende Eclipse poterlo utilizzare come editor di applicazioni PHP.

Directory: /net.sourceforge.phpeclipse_1.1.8.bin.dist.zip

PHTML ENCODER 4.1

PER CRIPTARE LE VOSTRE PAGINE PHP

Avete necessità di proteggere i vostri script PHP da occhi indiscreti? Ecco a voi un encoder che vi consente di codificare i vostri script prima di distribuirli. Poichè la tecnica usata è relativa proprio a PHP, i vostri script codificati funzioneranno su ogni piattaforma sia Windows che Linux.

Directory: /phtmlenc41.zip

PYTHON 2.5

L'EX GIOVANE RAMPANTE

Python è stato considerato per lungo tempo il nuovo che avanza. Attualmente non lo si può più definire in questo modo, Python è ormai un linguaggio stabile e completo che trova applicazione in un gran numero di progetti. Se ne parla sempre di più in campo industriale come su Internet. Soprattutto un gran numero di applicazioni anche in ambiente Windows girano ormai grazie a Python e presentano interfacce grafiche ottimamente strutturate. Ciò nonostante Python rimane un grande linguaggio di scripting adatto a gestire in modo completamente automatico buona parte di un sistema operativo sia esso Linux o Windows.

Directory: /python-2.5.exe

REGULATOR 2.0.3

PER NON AVERE PROBLEMI CON LE REGULAR EXPRESSION

Le Regular Expression rappresentano un odei sistemi più potenti che qualunque linguaggio porta in dotazione per i programmatori. Vengono utilizzate per effettuare ricerche su testi o stringhe SQL, per sostituire una stringa con un'altra e per molto altro ancora. Regulator è un tool per testare e costruire espressioni regolari in modo semplificato

Directory: /Regulator203.dotnet.1.1.zip

RUBY 1.8.5-21

IL VERO NUOVO CHE AVANZA

Se ne parla ormai come si parla di una rivoluzione. In America è il linguaggio che maggiormente ha scalato i vertici delle classifiche di utilizzo nell'ultimo anno. In Italia sta giungendo rapidamente come un ciclone a fare capolino nel panorama dello sviluppo. Bello, elegante, con una curva di apprendimento praticamente nulla, si tratta di un linguaggio di scripting che ottimamente si presta ad essere utilizzato sia sul Web sia in modo standalone. Recentemente ce ne siamo occupati in un bell'articolo di Paolo Perrotta, che mostrava come utilizzarlo per programmare un plugin per Skype messenger.

Directory: /ruby185-21.exe

SNIPPET COMPILER 2

PER COMPILARE UN PICCOLO PEZZO DI CODICE

Un utility che consente di scrivere, compilare e far girare piccoli spezzoni di codice. Molto utile se si vuole provare il funzionamento di una porzione di codice senza per questo dover creare un progetto completo con Visual Studio prima di eseguire tutto.

Directory: /SnippetCompiler2DotNet2.zip

STRUTS 1.2.9

IL FRAMEWORK MVC PER JAVA

Il pattern MVC è probabilmente il più usato da tutti gli sviluppatori in ogni linguaggio. Questo framework è il leader per quanto riguarda lo sviluppo di applicazioni Java secondo il pattern MVC. Molto comodo, stabile e affidabile, rappresenta la base di partenza per applicazioni che devono essere facilmente manutenibili. Uno standard da usare se progettate applicazioni di dimensioni generose, ma anche se volete sviluppare web application professionali

Directory: /struts-1.2.9-bin.zip

SUPEREDI 3.7

UN EDITOR PICCOLO E FUNZIONALE PER GLI SVILUPPATORI

Ideato appositamente per gli sviluppatori, Super Edi può essere utilizzato sia per lo sviluppo in locale sia per modificare file in remoto. Completamente gratuito, presenta tutte le principali funzionalità degli editor più blasonati: evidenziazione sintattica, filtri per la manipolazione automatica del testo e supporto multilingua.

Directory: /SuperEdi-3.9.U.exe

TOMCAT 6.0.2

L'APPLICATION SERVER PER JSP

Se siete dei programmatori JSP avete senza dubbio bisogno di Tomcat per testare e utilizzare le vostre applicazioni. E' senza dubbio un server wen ovvero può funzionare da server web ma è soprattutto un Application Server che vi consente di utilizzare la tecnica delle servlet o delle JSP per creare applicazioni Web con una logica business e con il linguaggio Java a fare da asse portante. Tomcat ha in se anche un Web Server, ma quel che più conta è un container di applicazioni Java. Potete e dovete utilizzarlo se volete imparare a programmare per il Web utilizzando il linguaggio di Sun.

Directory: /apache-tomcat-6.0.2.exe

VELOCITY 1.4

IL TEMPLATE ENGINE PER JAVA

Come si fa a separare la logica di business dall'interfaccia dell'applicazione? A risolvere questo problema arriva velocity. Scrivete l'applicazione, progettate l'interfaccia e velocity mette in comunicazione i due elementi. Utile e intuitivo rappresenta la soluzione ottimale quando non si vogliono usare mostri sacri come JSF.

Directory: /velocity-1.4.zip

CODICE BASATO SULL'EVOLUZIONE

LA PROGRAMMAZIONE EVOLUTIVA IDEATA DA L.J. FOGEL È UN METODO BASATO IN PARTE SULLA CASUALITÀ DEGLI EVENTI. IN QUESTO ARTICOLO ESEGUIREMO UNA CARRELLATA SUI PRINCIPALI STUDI EFFETTUATI IN QUESTO SETTORE

Dopo aver esplorato l'affascinante mondo degli algoritmi evolutivi e dei suoi principali derivati, ci apprestiamo a concludere il tour proponendo alcuni esempi applicativi concreti. L'obiettivo che ci poniamo, è superare, attraverso l'introduzione di un metodo del tutto generale l'approccio molto specifico con cui vengono affrontati molti problemi. Evidentemente si tratta di un tentativo molto ambizioso, ma che vale la pena perseguire poiché, come molti studi confermano, si incominciano a intravedere incoraggianti risultati. In particolare, quando si parla di affrontare in modo generale un problema, in questo ambito, si intende modellarlo sulla base delle strutture e le tecniche proposte dalla programmazione evolutiva. Consideriamo alcuni processi di progettazione ingegneristica in cui è necessariol'uso di strumenti ad hoc per modellare fenomeni fisici. Ora, il punto è che le variabili in gioco sono talmente tante che per ottenere delle variazioni di misura significative a seconda della situazione in cui il fenomeno fisico è innestato che per valutarle tutte bisognerebbe ricorrere a strumenti di calcolo parallelo, oppure considerare tempi di risposta estremamente lunghi.

Si percepisce immediatamente che soluzioni basate sul calcolo parallelo sono spesso antieconomiche o comunque improponibili a causa di motivi tecnici o semplicemente perché in ogni caso il volume di calcolo è talmente grande che anche l'impiego di architetture particolarmente sofisticate renderebbe difficile l'esecuzione in tempi utili alla ricerca.

Un primo miglioramento è stato segnato dall'introduzione degli algoritmi genetici (GA – genetic algorithms) ed in particolare della variante, o come alcuni chiamano del dialetto, della programmazione evolutiva (EP – evolutionary programming). In questo settore si sono ottenuti molti progressi anche se a tutt'ora la ricerca è ancora lontana dall'essere perfetta.

SELF ORGANIZED CRITICALY

Self organized criticality conosciuto con il suo acronimo SOC è un modello intorno al quale si è registrato in questi anni un'intensa riflessione. In molti campi mediante questa teoria, che potremmo tradurre come, "*criticità auto organizzata*" si sono affrontati svariati tipi di problemi. Ma di cosa si tratta? Con SOC si tenta di dare una risposta alla domanda sempre più frequente di previsione del comportamento di alcuni fenomeni "poco trattabili", ossia fenomeni che ad esempio mostrano elevata complessità e comportamenti fortemente non lineari. Si tratta del metodo più accreditato per affrontare problemi caotici descritti da modelli particolarmente complessi. Per sistemi in equilibrio stabile la teoria è ormai consolidata poiché perturbazioni all'assetto di equilibrio provocano deviazioni proporzionali alla intensità del disturbo immesso. E così simili sistemi sono facilmente associabili a opportune funzioni obiettivo. In sistemi caotici anche piccoli disturbi possono provocare razioni violente del sistema in modo da amplificarne le

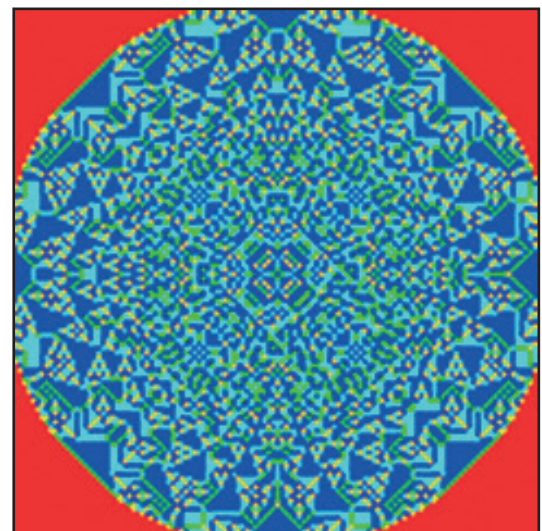



Fig. 1: *"Pila di granelli di sabbia (sandpile) simulata con una funzione frattale"*

risposte in modo a volte esponenziale. I sistemi caotici spesso non presentano memoria dei loro stati passati, per cui è estremamente difficoltoso prevederne l'evoluzione. Nella transizione dallo stato stabile a quello caotico possono emergere comportamenti complessi. I sistemi di criticità auto organizzata evolvono verso lo stato critico senza alcuna forza esterna organizzatrice. Questo è un vantaggio poiché non è richiesta a priori alcuna informazione circa le funzioni interne che descrivono il sistema. Il classico esempio per descrivere i SOC e la pila di granelli di sabbia (sandpile). In questo modello si dispongono in pila una notevole sequenza di granelli di sabbia. La struttura si auto organizza con caratteristiche dinamiche, si possono registrare reazioni di semplice rotolamento di pochi granelli come molto più cospicui scivolamenti valanghe di parte della sabbia. Il sistema, autonomamente, tende a raggiungere uno stato di criticità.
Un sistema che reagisce in questo modo agli input esterni si dice *power law*. Basato cioè su una legge che lega più componenti di uno stesso sistema. In particolare si ha quando una componente si può esprimere in funzione della potenza (in termini fisici) di un'altra componente. Così le varie componenti del sistema sono tra loro relazionate da una fitta e complessa rete di funzioni tra loro correlate. Negli ultimi anni si è potuto riscontrare come molti eventi seguono tale legge: dalla distribuzione dell'energia dei fenomeni sismici, alle popolazioni delle città; per passare a problemi di linguistica computazionali e per finire alle geometrie della natura (corsi di fiumi e coste di monti). Molti modelli matematici, altamente eterogenei tra loro, sono assimilabili a tale legge: dai frattali alle leggi di Zipf e Pareto e alla modellazione del rumore con 1/f. Gli studi di Bak, il pioniere della SOC hanno evidenziato che lo stato critico è il più efficiente che si può raggiungere dinamicamente. Nei fenomeni naturali l'ottimimizzazione è il processo che il sistema compie per adattarsi alle condizioni esterne (ad esempio cambi di pressione). Bak sviluppo un modello che descriveva l'evoluzione della specie come una sequenza di due singoli transizioni: interazione e selezione. Avendo inizialmente posizionato casualmente un numero di individui della popolazione in un "campo" con topologia ad anello; ad ogni passo l'individuo in esame e i suoi due vicini (adiacenti) sono sostituti con una nuova specie secondo regole che hanno anche una componente casuale. Secondo un modello simile a quello proposto da Stephen Wolfram negli automi cellulari unidimensionali. Si è dimostrato che il complesso comportamento evolve gradualmente con miglioramenti del fitness della popolazione.

UN'EVOLUZIONE DI SOC CON EP

Il passo successivo è stato quello di introdurre la programmazione evolutiva EP in modo da migliorare i modelli SOC. Così è nato EPSOC. La formulazione del generico problema di ottimizzazione con un singolo obiettivo consiste nella minimizzazione di una funzione f(x), da R^n a R generalmente non lineare con x= $\{x_0, x_1, \ldots x_n\}$ dove il generico x_i appartiene a R. Una interpretazione secondo il calcolo evolutivo, considera la popolazione p come un set di vettori parametri, ossia p= $\{x_0, x_1, \ldots x_m\}$. Nelle effettive applicazioni ingegneristiche ad esempio i valori di f(x) generalmente derivano dalla esecuzione di complesse simulazioni numeriche che richiedono un significativo tempo di computazione. Si può descrivere l'algoritmo EPSOC come una sequenza di passi:

1. Inizializzazione random, uniformemente distribuita, della popolazione p e valutazione di ogni possibile soluzione x_i, per tutti i possibili valori di i;
2. Ordinamento della popolazione secondo i valori della funzione obiettivo f(x);
3. Selezione di un insieme B, dei "peggiori" membri della popolazione, sia *np* la cardinalità di tale insieme. Per ogni membro di B, si aggiungono all'insieme i vicini (adiacenti) qualora non appartenessero a B;
4. Nuova inizializzazione applicando una mutazione random a uniforme distribuzione all'in-

STORIA DEL CALCOLO EVOLUTIVO

La macro area in cui sono collocati tutti gli studi riferiti all'automazione della teoria della selezione naturale darwiniana è conosciuta come *calcolo evolutivo* EC o *algoritmi evolutivi* EA. Negli *algoritmi genetici* GA i membri delle popolazioni sono rappresentati da stringhe di bit che chiameremo geni. Su di essi si applicano operazioni di crossover, di mutazione e di selezione mediante la sostituzione dei padri con i figli. Le *strategie evolutive* ES sono applicate a reali problemi di ottimizzazione storicamente discretizzati. Qui il principale operatore di evoluzione è la mutazione (come funzione gaussiana) che cambia stocasticamente alcuni parametri della funzione obiettivo. Storicamente non è presente la riproduzione come crossover. La *programmazione evolutiva* EP si è concentrata sull'evoluzione simulata come l'azione di un automa a stati finiti. Si iterano tre passi fondamentali: inizializzazione pseudo casuale della popolazione, riproduzione con crossover e mutazione e valutazione del fitness al fine di sostituire una nuova generazione con parte di quella esistente. Questo ultimo punto è un punto molto delicato per la EP e va attentamente valutato per ottenere risultati soddisfacenti. Si investiga, quindi molto sul rapporto padre figlio. La *programmazione genetica* GP, vuole realizzare una *learning machine* capace di produrre in modo automatico programmi. Il crossover e la mutazione operano su un albero con dimensioni dinamiche.

ARTICOLI CORRELATI

Gli articoli scritti da Fabio Grimaldi nella sezione soluzioni a cui fare riferimento per eventuali approfondimenti sono: Il gioco della vita - n. 34; Automi cellulari unidimensionali - n. 53; Modelli di contagio con automi cellulari - n.54; Un automa cellulare doppio - n.55; La serie sui frattali - n. 50, 51, 52; Costruzioni geometriche iterative - n. 85; Sistemi caotici - n. 86; sistemi di Lindermayer - n. 87;. La serie sul calcolo evolutivo n. 108, 109,110.

sieme B. Tutti gli altri membri della popolazione saranno usati per la riproduzione. I figli saranno generati usando una piccola mutazione (circa il 10% del sotto insieme considerato) casuale e uniformemente distribuita mutazione dei genitori.

5. Valutazione di ogni nuova soluzione f(x);
6. Se il figlio ha una migliore funzione obiettivo del padre allora si sostituisce il padre con il figlio;
7. Ripetizione del passo 2 finché uno stabilito numero di iterazioni non sono state eseguite.

Un primo punto debole potrebbe essere la valutazione della funzione obiettivo che richiede un consistente sforzo computazionali. Il problema può essere superato con il calcolo parallelo. Questo metodo è stato usato in molte applicazioni ingegneristiche ed a mostrato buone performance. In molti casi si è mostrato più efficiente degli algoritmi tradizionali.

OTTIMIZZAZIONE MULTI-OBIETTIVO

In problemi del mondo reale, spesso capita che si intendano raggiungere più obiettivi. Il classico esempio è quello riferito alla produzione industriale dove si vogliono massimizzare le prestazioni del prodotto finito da produrre e simultaneamente si vogliono minimizzare i costi. L'approccio più comune che la ricerca operativa ha nel passato proposto è quello di considerare un'unica funzione obiettivo e molti vincoli con i quali si cercava di soddisfare gli altri obiettivi. Con gli algoritmi evolutivi EA è stato concepito un nuovo modo di agire che permette di valutare problemi con più obiettivi, definendo quindi un multi obiettivo. Alcuni obiettivi possono essere in competizione tra loro e quindi necessaria una scelta. In linea di principio esistono tre metodi per farlo che si distinguono per il timing della decisione rispetto alla ricerca:

- Decisione prima della ricerca: si aggregano gli obiettivi in un singolo obiettivo. Il problema si riduce ad una ottimizzazione a singolo obiettivo.
- Decisone durante la ricerca: in modo interattivo si sostituiscono le preferenze e così viene guidata la ricerca.
- Decisione dopo la ricerca: viene selezionata la soluzione candidata dopo la ricerca.

Il più innovativo, perché non degenera in metodo già conosciuti è il terzo. Applicando EA a problemi di ottimizzazione multi obiettivo è necessario definire due principali problemi. Il primo è trovare una maniera per assegnare il fitness e per elaborare la selezione. Il secondo riguarda, a fronte di diverse popolazioni e dello spazio di ricerca, il modo di affrontare la ricerca. L'assegnazione del fitness può essere affrontata in diversi modi:

- Basata sull'aggregazione;
- Basata su criteri;
- Basata sul criterio di Pareto

Una soluzione che ha fornito risultati apprezzabi-

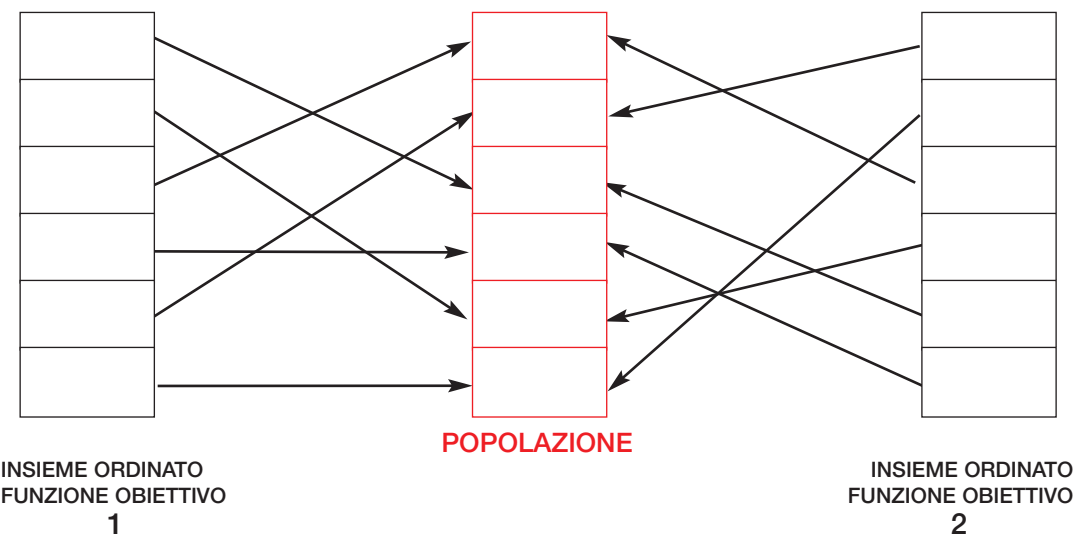



Fig. 2: Tipica struttura dati di EPSOC con due obiettivi"

POPOLAZIONE

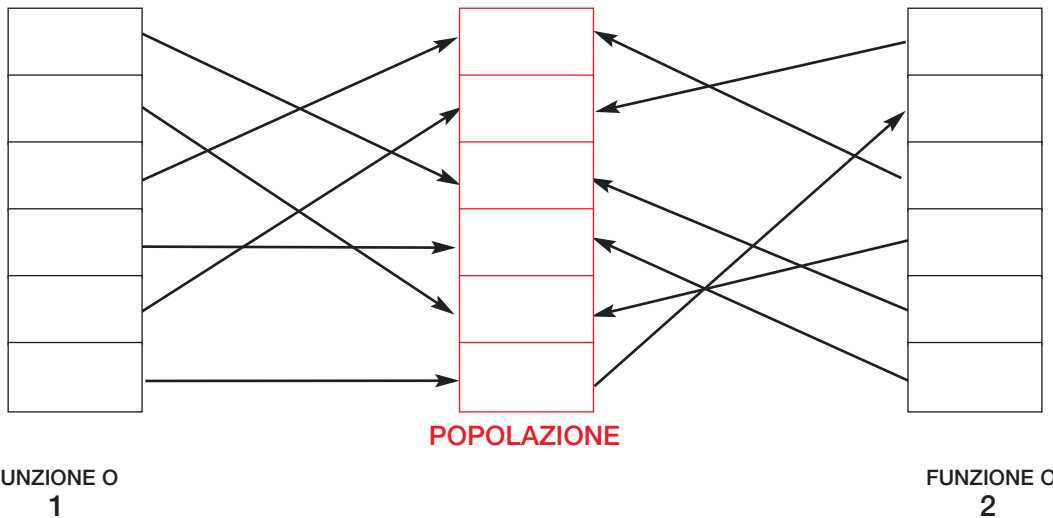
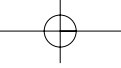
FUNZIONE O 1

FUNZIONE O 2

Fig. 3: Mappatura la contrario per EPSOC

li è un mix tra quella basata sul criterio e quella basata su Pareto. Il fitness a turno è definito dai valori delle funzioni obiettivo, basata sul criterio. Ma la selezione per estinzione è indicata da una graduatoria di tutti gli obiettivi, un esplicito riferimento al metodo Pareto. La diversità della popolazione si ottiene allo stesso modo indicato per il SOC a singolo obiettivo, ossia attraverso l'accoppiata di mutazioni ed estinzioni. L'estinzione è la conseguenza della nuova inizializzazione ottenuta da una accurata selezione della popolazione. EPSOC opera su una popolazione ordinata per fitness in relazione alla funzione obiettivo. Se gli obiettivi sono più di uno, un modo per rappresentare l'intera struttura ordinata, prevede l'uso di tanti vettori quanti sono gli obiettivi, tutti dinamicamente linkati alla popolazione, come mostra la **figura 2**.
Emerge come i membri da scartare (denominati nell'algoritmo i peggiori) per un obiettivo, possono essere invece per un altro obiettivo tra i migliori, quindi da non scartare; e viceversa. Utilizzando una simile struttura dati, ma con link al contrario, si possono gestire le situazioni per le quali un membro della popolazione è una élite per un obiettivo mentre e tra i peggiori per un altro. In tal caso ad esempio si può decidere di non estinguerlo in conformità al criterio di Pareto. Se accade come in **figura 3** che l'elemento peggiore della funzione 1 è tra i migliori della funzione 2 allora si può decidere di non estinguerlo. Ovviamente va definito un criterio, generalmente si usa un ranking, ossia un punteggio per ogni membro. Analogo discorso può essere esteso ai vicini di ogni individuo della popolazione (passo 3 dell'algoritmo). Un altro modo per trattare la questione fa uso di archivi che garantiscono l'ottimalità del criterio di Pareto. Si salvano i membri che, per così dire, garantiscono ottime performance.

ORIENTARSI NEL CALCOLO EVOLUTIVO

L'idea di applicare le rivoluzionarie scoperte di Darwin alla automazione e alla programmazione ha aperto una strada che aspetta ancora di essere percorsa a pieno. Negli Anni Sessanta sono emersi i primi studi che se pure si basavano sulla teoria della selezione naturale approntavano diversi approcci. Negli Stati Uniti Fogel introduceva la programmazione evolutiva - EP , e Holland sviluppava gli algoritmi genetici - GA. Negli stessi anni in Germania Rechenberg e Schwefel partorivano le strategie evolutive - ES. Gli studi sono proseguiti separatamente per una quindicina di anni fin quando non è stato tentata un'unificazione di tali studi che molti consideravano già diversi dialetti di una stessa lingua. Veniva costituto il calcolo evolutivo EC o algoritmi evolutivi – EA. Per cui tutti gli approcci di studio sono oggi considerati dei derivati di EA. Gli studi di Koza circa la "learning machine" dei programmi, conosciuti come programmazione genetica - GP, sono da ritenersi più recenti, così come i sistemi classificatori e la swarm intelligence –SI. Partiamo dalla radice. L'idea comune a tutte le tecniche è la legge di selezione naturale darwiniana. Data una popolazione di individui, l'ambiente causa la selezione naturale. Viene definito un parametro che è il (o se preferite al femminile la) fitness che è un sorta di idoneità dell'individuo rispetto all'ambiente, si tratta di una stima della probabilità che esso viva abbastanza da riprodursi. Se l'intera popolazione "evolve" il fitness riferito ad essa cresce. Chiaramente emerge una stretta relazione tra un problema così formulato è un processo di ottimizzazione. Data una funzione obiettivo da massimizzare si possono individuare casualmente (o pseudo casualmente con procedimenti in parte casuali e in parte euristici) le funzioni candidate e una volta misurato il fitness se-

lezionare la migliore. Ma i riferimenti alla selezione non finisco qui. Oltre al riferimento alla selezione un'altra fase del processo prevede di produrre nuove soluzioni da quelle scelte come processi di riproduzione (ottenuti dalla combinazione di due soluzioni che genera nuove soluzioni "figlie") o mutazione (ottenuti dal cambiamento di una sola soluzione che ne produce una nuova). Questi processi che generano della progenie possono essere iterati al fine di migliorare continuamente il fitness dell'intera popolazione. È importante rilevare come alcuni anelli di tale catena sono legati a processi stocastici, così come avviene in natura secondo Darwin. In questo modo si spiega l'adattamento degli individui all'ambiente che nel tempo si può alternare. La stima del fitness solitamente è approntata con un'euristica. La riproduzione e la mutazione sono i due procedimenti che garantiscono la selezione naturale. Da un lato tendono a migliorare la popolazione dall'altro assicurano la varietà della specie. In un processo artificiale vanno attentamente costruite a seconda dell'obiettivo che si intende raggiungere. Un secondo approccio potrebbe essere quello dell'evoluzione naturale dove si riscontra un processo ciclico molto simile a quello appena descritto. In particolare usando il linguaggio della biologia si può dire che l'evoluzione naturale agisce sul materiale genetico, conosciuto come genotipo, di un individuo anziché sulle sue caratteristiche fisiche, il fenotipo. Ogni variazione che promuove l'adattamento di un individuo emerge dal patrimonio genetico, e come Darwin ribadisce, non dall'apprendimento o dalle esperienze che i genitori hanno registrato nel corso della vita. La selezione naturale favorisce la riproduzione degli individui che migliorano l'adattabilità all'ambiente alterabile ed elimina gli individui dalla minore potenzialità riproduttiva. Dal punto di vista genetico, la selezione naturale promuove quelle particolari combinazioni genetiche che danno vita a un organismo più efficiente, selezionando il genotipo, non il fenotipo. La riproduzione è l'elemento fondante del processo evolutivo: la variabilità generazionale di una specie è determinata dalla ricombinazione genica (tra due genitori) e dalle piccole mutazioni casuali del codice genetico (su un singolo individuo). Si determinano così differenze tra figlio e genitori. La variabilità è una condizione essenziale dell'evoluzione. L'evoluzione naturale opera su intere popolazioni attraverso processi ciclici e generazionali determinati dalle contingenze ambientali e dalle interazioni fra i vari organismi. L'inizializzazione è un processo stocastico attraverso il quale si introducono degli individui all'interno dello spazio di ricerca. In taluni casi si usa anche l'inoculazione, ossia l'inserimento forzato di individui con caratteristiche prestabilite. La valutazione del fitness è di fatto il maggiore sforzo computazionale che si attua. Per questo spesso si utilizzano architetture parallele per eseguire questo passo.
La riproduzione avviene spesso mediante crossover, esso deriva dalla sintesi delle due parole "crossing over" è il modo in cui vengono combinati tra loro i geni dei due genitori per la produzione di geni figlio. Esistono molti modi per simulare tale "mescolamento". Modelli sofisticati sono in grado di tenere conto di molte delle informazioni contenute nella lunga catena cromosomica. Ad ogni modo, per intenderci e capire come può essere simulata tale fase, possiamo esaminare il basilare metodo di "single point crossover". Con tale tecnica le due catene dei genitori vengono tagliate alla stessa lunghezza formando cosi entrambe due tronchi: una testa ed una coda. Si tratta a questo punto di produrre due nuovi geni figlio. Il primo con una testa di un genitore e la coda dell'altro e il secondo con i rimanenti due tronchi. Le mutazioni sono alterazioni stocastiche della catena genica. Normalmente si prevedono piccolissimi (dell'ordine di frazioni dell'uno percento) cambiamenti. Una variazione sul tema della mutazione è l'inversione per la quale gruppi di bit della catena cromosomica del gene vengono invertiti come not logico.

CRITERIO DI PARETO

Questa famosa e importante legge deriva da uno studio che Vilfredo Pareto fece alla fine del 19° secolo riguardo alla distribuzione dei redditi in una regione. Egli dimostrò che una minoranza della popolazione possedeva la maggioranza della ricchezza. Venne introdotto il famoso rapporto 80/20, riferito appunto alla ricchezza e alla popolazione.
Da allora si indica questo criterio quando il l'ottanta per cento delle risorse è assegnato al venti per cento dei richiedenti. Ovviamente i due numeri sono soltanto indicativi.

CONCLUSIONI

Il mondo degli algoritmi evolutivi si dimostra decisamente complesso. Il tentativo è quello di incapsulare in algoritmi al cui interno sia inserita una forma di "casualità controllata" quello che è il tipico comportamento degli elementi naturali. Nonostante lo sforzo sia estremamente gravoso i risultati fin qui ottenuti si dimostrano particolarmente incoraggianti. La dove il problema lo richieda è particolarmente stimolante il provare ad inserire questi algoritmi nel vostro codice

Fabio Grimaldi